Educational Technology/September-October 2004, pg 10-16

Design Tradeoffs in Usability and Power for Advanced Educational Software Authoring Tools

Tom Murray

www.tommurray.us

Authoring tools for advanced learning environments. Despite many years of research and development, advanced or adaptive learning environments (ALEs) such as intelligent tutoring systems, adaptive hypermedia systems, and coached inquiry leaning environments have seen relatively little use in schools and training classrooms. This can be attributed to several factors that most of these systems have in common: high cost of production, lack of widespread convincing evidence of the benefits, limited subject matter coverage, and lack of buy-in from educational and training professionals. Authoring tools are being developed for these learning environments because they address all of these areas of concern (Murray et al. 2003). Authoring tools can reduce development time, effort, and cost; they can enable reuse and customization of content; and they can lower the skill barrier and allow more people to participate in development and customization (Ainsworth et al. 2003, Half et al. 2003, van Joolingen & de Jong 1996, Munro et al. 1997, Merrill 2003). And finally, they impact LE research, evaluation and evolution by allowing alternative versions of systems or instructional strategies to be created more easily, and by allowing greater participation by teachers and subject matter experts in the R&D process.

Much progress has been made in the last two decades developing authoring tools for advanced LEs. A recent edited book contains chapters describing 16 such systems (Murray et al. 2003), and includes an overview chapter by myself comparing and

contrasting 28 authoring systems (Murray 2003) (see http://helios.hampshire.edu/~tjmCCS/atoolsbook/). Still, the field is in an early stage and authoring tools are far from "saving the day" by allowing subject matter experts (SMEs--which includes teachers and curriculum developers for the purposes of this article) to create highly adaptive tutors for their domains. In this article I will outline some issues that I hope will shed light on what we can expect from authoring tools, given the design tradeoffs involved.

Who authors?

Advanced and/or adaptive learning environments are those computer-based LEs that include advancements over traditional courseware, multimedia edutainment, and educational simulations by having any of the following characteristics : 1) They go beyond multiple choice questions or hyperlinked web pages and include complex interactions, open ended problem solving, and flexible information and activity spaces. 2) They go beyond simple sequencing and branching of content to heuristic selection or sequencing of topics, activities, and feedback based not only on the single prior student behavior, but based on some processing of the history of student behavior (as in a student model). The additional flexibility and complexity has the potential to enable much richer and more authentic learning contexts, and to provide feedback more closely tailored to the student's need, but all this comes at the cost of increased complexity. Most ALEs contain knowledge or information in several distinct categories, such as activities, topic networks, student model parameters, subject matter knowledge, teaching strategies, etc., thus and ALE "authoring tool" many be composed of several authoring tools, one for each type of information.

There is some debate in the field as to whether authoring tools should be tailored to be used by practicing classroom teachers (or SMEs) or by "knowledge engineers" who specialize in creating formal and computational representations of subject matter and

instructional knowledge (or both). The target user audience in large part determines the complexity of the authoring tool. In the next decade will undoubtedly see be a wide range of complexities among LE authoring tools, but one thing we can confidently claim is that SMEs have to play some part in the LE design process. Some might say that the role of SMEs can be kept to a minimum. But principles from human-computer interaction theory are unequivocal in their advocacy for continuous, iterative design cycles in authentic user and task contexts (Shneiderman 1998, Norman 1998). These principles also advocate for participatory design in which end users participate intimately in the design process as consultants or co-designers, not just as evaluation subjects. This leads to two conclusions. First, LE design requires iterative testing with students. This in tern requires the participation of SMEs (with expertise in the domain and with teaching), as only they can provide insightful evaluative analysis that points to whether a LE is achieving its goals, and, especially, why it might not be if there are problems. In Murray (2004) I give a more detailed argument for why SMEs need to participate deeply in LE design, with an example from authoring an inquiry-based medical diagnostic tutor.

The second conclusion is that, since authoring tools need to be usable by SMEs, then SMEs need to be highly involved in the formative stages of designing the authoring *tools* themselves. This is essential in order to insure that authoring systems can in fact be used by "average" (or even highly trained) SMEs.

Of the many research publications involving authoring tools, extremely few document the use of these tools by SMEs not intimately connected with the research team (exceptions include work described in (Ainsworth et al. 2003) and (Halff et al. 2003)). Current systems give the impression of being quite complex, and the technology transfer from research lab to authentic use has not been sufficiently mapped out. Even given the wide gap in empirical evidence for LE authoring tool usability, some general claims about the tradeoffs involved in designing usable authoring tools can be made which will allow us to have appropriate expectations about their potential.

Complexity and usability

As mentioned LEs tend to be quite complex. Authoring tools relieve the designer of the task of programming by providing suitably generic and flexible building blocks and customization parameters, but authoring LEs is still a formidable task. Due to the tradeoffs involved, there are essentially two options for making authoring easier: 1) use an authoring tool specially tailored to the type of LE that you want to build, or 2) limit what you intend to build to be fairly simple, and use a simple authoring tool. But if you want an authoring tool that can build complex tutors in many domains, usability becomes more problematic. We have no canonical theory for the design of LEs (indeed, there are no universally agreed upon strategies for structuring or delivering instruction, and there may never be) so there are many ways to design a tutor for a given set of topics, and numerous design decisions to be made at many levels.

One size does not "fit all"-- multiple authoring roles. The tradeoffs between ease of use and powerful features is now widely understood as consumers have more options for their commonly used computer applications such as spread sheets, graphics tools, and web site authoring tools. As an analogy, assume that you are given software to design a new home (you don't have to worry about constructing it--the contractors will do that for you). One can image a range of such authoring tools, from one that specializes in designing fully equipped Tudor-styled eight bedroom houses by choosing preferences from menus (the specialized tool), to one that allows you to design almost anything as long as it is only one room (the simple shack tool), to one that allows you to design any house but is so complex only an architect could understand how to use the authoring tool. One partial solution is to support different user levels of complexity, or to hide complex features, to allow both ease of use and powerful features (for example, by putting common tools in a tool bar and hiding advanced tools deeper in menus). This works to some extent, but building complex systems involves complex design thinking. So multi-user-level tools can allow a novice to design a simple artifact and an expert to

design a complex artifact, but can not support a novice to design a complex artifact (unless it has templates for specific types of LEs, which is the "special purpose" category).

The optimal solution is one that allows collaborative design with multiple roles. In the case of the architectural software example, we would want a tool that an architect could use to design drafts to be viewed and commented on the by customer. The tool would also support the customer in visualizing the design, understanding various options and their implications, and in other ways becoming intimately involved in an iterative design process, perhaps allowing simple modifications such as color selections or moving appliances around. Similarly with LE authoring systems, tools should be powerful and usable enough for both knowledge engineers and teachers to participate (and SMEs, in the case where the SME is different than the teacher). To summarize, assuming as argued above that the teacher/SME needs to be involved in the process, the usage scenarios are as follows:

- Use a special purpose authoring tool (but: it has limited flexibility, available ones may not fit your needs)
- Use a simple tool (but: it has limited power and depth; you may be stuck with multiple choice interactions, etc.)
- Work with a trained knowledge engineer (but: can be hard to find, and expensive)
- Provide advanced training to the teachers/SMEs (but: is expensive and time consuming)

Of course good interface design and appropriate representational formalisms also make a big difference in usability, and must be considered in authoring tool design. There are no magic bullets here, as the above list suggests. And of course, powerful authoring tools are not yet even available to the average educator. But the situation is not so bleak. Both special purpose tools and more generic tools could be on the market within a few years,

and the constraint of hiring one trained knowledge engineer per college, public school district, or company (option #3 above) is far from prohibitive. Such experts would work with SMEs, configuring, consulting, and training on an ongoing basis.

Authoring tool design tradeoffs

The following list illustrates the full range of tradeoffs involved in designing an authoring tool.

- Power / Flexibility
  - Breadth (scope)
  - Depth
- Usability
  - Learnability
  - Productivity
- Fidelity
- Cost

These are "tradeoffs" because trying to maximize one dimension usually comes at the cost of another. Overall, Power/flexibility and Usability are usually at odds with each other, since simplicity tends to correlate with usability. **Power/flexibility** has two aspects: breadth (scope) and depth of the content. **Breadth** is how general the framework is for building LEs for diverse subject areas and instructional approaches. Knowledge **depth** is the depth to which a system can reason about or teach the knowledge. Breadth and Depth are often at odds, because one must often limit the generality of a system to be able to represent deep causal knowledge. **Usability** also has two aspects: learnability and productivity. **Learnability** is how easy a system is to learn how to use. **Productivity** is how quickly a trained user can author a LE. Learnability and productivity are often at odds, since a system that is designed to be picked up quickly by novices may not provide the powerful features that experienced users need to efficiently produce large systems.

Fidelity is the degree to which the authored LEs can perceptually and operationally match the target domain. A 3-D immersive environment has more visual fidelity than a 2D simulation. A learning environment that allows the student to directly

practice a task has more fidelity than one that merely describes and asks questions about the task. **Cost** refers to the resources needed to build the authoring system (for this discussion the availability of personnel, expertise, and time are included in this category). The cost of building each LE itself depends on the other factors (usability, etc.), and of course on individual properties of the LE such as its size. "**Effectives**" of the LE, i.e. its success in supporting learning, is of course an important factor, but this factor depends mostly on the design of each individual LE, and not as much on the design of the authoring tool, so effectiveness is not included in the list of tradeoffs. Finally, the design tradeoffs may come into play for each authoring tool component separately. For example, an authoring system can have a highly usable tool for authoring shallow domain content and a not so usable tool for authoring complex teaching strategies.

Towards estimating complexity

Informal comparison with familiar applications. In parallel with accumulating empirical evidence for the usability of authoring tools, we can engage in a "rational analysis" that will help us predict what types of skills or levels of training are needed to use certain authoring tool features. That is, given a complex authoring environment, an analysis of its features should theoretically provide some initial guidance on usability (or, in the other direction, this analysis can constrain authoring tool design according to intended user skill level). ALE authoring systems are novel, but we can draw some parallels with existing software and educational design tasks. For example, we can analyze the authoring process required by a particular tool and ask to what degree is it similar to:

- Authoring a book or curriculum? (Or simply planning a class?)
- Using a word processor, or Excel spread sheets with macros, or doing data base design and maintenance?
- Formalizing one's tacit knowledge of a subject or teaching strategy (perhaps

similar to drawing concept maps or writing expert system rules)?

In other words, we can analyze components of the LE authoring process and compare their complexity with that of more familiar computer applications. We can then ask: how much training and practice does it take for most people to learn how to use a similar application. Perhaps unfortunately, most existing advanced LE authoring tools fall on the complex side of the items in the list above. The authoring task usually requires an ability to conceptualize and structure the concepts and skills in a domain from a high level.

Components of complexity. We can move toward a scheme for quantitatively estimating authoring tool complexity with the following list of authoring tasks of increasing complexity, from properties, to objects which have multiple properties, to structures connecting multiple objects, to models that relate objects and structures over multiple time slices:

1. **Properties**--authored with simple interface choices and data entry. Text boxes, sliders, lists, etc.
2. **Objects**--authored with templates, forms, or schemas. The user must understand how the properties of an object work together to define or characterize objects (concepts, skills, media content items, hints, examples, links, etc.).
3. **Structures**--multiple relationships and "big picturing". Users have to define relationships between objects, and conceptualize beyond the local relationships to more global patterns or properties in a network of objects.
4. **Modeling and behaviors**. When the relationships between objects include "IF/THEN" conditionals or cause & effect links, the structures become dynamic models. This leads to the issues listed under "dynamic complexity" below.

As indicated above, authoring involves keeping track of numerous interrelated "objects", a task not unlike database management. And authoring usually involves a metacognitive or reflective level of understanding of how and why one chooses among various instructional tactics. The complexity of the authoring task can be further elaborated from the list above:

1. **Interface and tool complexity**: When there are many features and tools, it becomes difficult to manage them. Each object is like a building material, and each feature is like a tool. The advanced level user must learn how to use and orchestrate many tools to combine many objects.
2. **Object complexity**: LEs often involve abstract concepts who's definitions and uses are not obvious. For example it may be necessary to have a clear distinction between concepts, skills, and principles, if these are among the building blocks used. These objects are not as easy to grasp conceptually as the more concrete objects such as "rectangles," "lines," "paragraphs," and "fonts" in other applications.
3. **Structural complexity**: LEs often include complex structures of linked objects, as in student models, teaching models, or subject matter expert system models. Creating and maintaining such relationships and structures is cognitively challenging compared to the tasks required by most applications.
4. **Dynamic complexity**. LEs are artifacts with behaviors that can be "run" to test them. Advanced LEs have many possible student paths, and it becomes intractable to test every possible student behavior. Unpredictable behaviors can result. It is not unlike the task of writing and debugging a computer program, which takes special skill. If, during a test run with a student, an undesirable software behavior is noticed, it can become quite difficult to determine where to look to locate the cause.

We can characterize authoring tool complexity through a separate analysis at each of these levels. At the interface and tool level it may be sufficient to simply count the number of properties and objects involved in a average LE or tutorial, and the number of tools needed to author it. At the Object level, we can count how many abstract objects or elements are used. At the Structural level, one-to-one mappings are the simplest, one-to-many mappings are more difficult, and many-to-many mappings are most complex to manage and conceptualize. An example of a one-to-one mapping is if each learner task had one and only one graphic picture associated with it, and each graphic is used in only one place. An example of a many-to-many mapping is where each topic can be taught with multiple examples, and each example might be used for multiple topics. Structural interdependence and constraints can also create complexity when changing one value will change, or require the author to change, other values. At the Dynamics level, we can note that procedures or scripts with branches are relatively easy, procedural loops are more difficult to create and debug, and parameterized or recursive procedures are the most difficult.

The unfortunate fact is that most of us can not name many software users who are not software professionals who use software with a very high object complexity, structural complexity, or dynamic complexity. This points again for the need of knowledge engineers on design teams. It also provides some qualitative metrics that might allow us to design authoring tools for particular targeted users.

Quantifying tool complexity. A quantitative analysis of authoring tool complexity might include the following. I list properties of the tool, followed by properties that can be estimated for building a LE of average size using the tool.

Tool Properties:

- number of tools (or ways to manipulate objects and information)
- number of object types
- number of abstract object types (e.g. “graphic” is concrete, “topic” is abstract)

- number of object properties (totaled over all object types)
- number of constraint relationships (changing one thing changes other things)
- number of one to many, one to many, and many to many relationship types allowed
- procedural scripting level: branching, loops, parameterized procedures, recursion.

LE Properties:

- Total number of objects
- Total number of object relationships
- Total one to many and many-to-many relationship

This type of analysis could be done in several ways. One could make two counts, once for the "basic" or commonly used features, and once for all features. Also, the LE Properties could be calculated for several types of authored LEs, for example small, medium, and large (or simple and complex).

From complexity metrics to usability metrics. We need to be concerned with several aspects of usability. In addition to the conceptual and interface types of complexity indicated by the list above is the *familiarity* of the conceptual structures and mental models (or "epistemic forms" from Collins & Furgeson 1993) involved in using the tools. Forms, models, and procedures that are familiar from using other software or from common use in a subject matter area will be easier to learn and use. There are no quantitative methods for measuring familiarity or the esoteric-ness of the mental models involved, but a "low, medium, high" rating could be devised for comparative purposes.

The above scheme would allow a rather high level of precision for a multi-dimensional metric of authoring tool complexity (which would be a main factor in its usability).

However, the effort to document tool complexity to this level of detail will not be necessary for many situations. A full characterization is appropriate if one is trying to compare two similarly complex tools, or if one is trying to gauge the amount and type of authoring tool training needed for a particular user audience . However current authoring tool projects have done almost no analysis of this type, and a "quick and dirty" use of the scheme would yield useful results. For example, I believe that a rough analysis of this type would clearly indicate that many authoring tools projects overestimate the skill level needed for their target audience, or underestimate the amount of training needed. For precise use of the scheme to order the complexity of several authoring tools one would need to create a total complexity or usability metric that collapses the dimensions of variation into a weighted total of all of the factors. We do not suggest one here because at this stage of the work the details of the weighting would be rather arbitrary. Also such a total score may be of limited use. A realistic comparison of two systems would look at how they compare across several dimensions and consider which dimensions are most important for a particular author audience.

Reducing Complexity and Hidden Complexity

Some design strategies for making tools more usable follow directly from the above discussion. Liming the number or complexity of objects, relationships, and advanced tools, while making common and easy tools more salient, for example. Beyond the scope of this article are important considerations and principles for interaction, layout, color use, etc. found in the interface design literature (Shneiderman 1998). However even when it seems that we have designed a relatively simple and usable interface, we must also be aware of "hidden complexity." Adaptive learning environments tend to be rather complex, and designing highly usable LE authoring tools often involves automating certain processes or inferences. (In Murray 2003 I describe a number of knowledge acquisition methods used in authoring systems: including scaffolding, defaults,

visualization, elicitations, validation, and automation.) The automated stuff done "behind the scenes" to make the author's job easier can come back to haunt him in later phases of development. I will use the Redeem authoring tool for intelligent tutoring systems as an example (Ainsworth et al. 2003).

An example of hidden complexity. Redeem is designed to allow authors to re-use multimedia content that was originally created for non-intelligent computer-based training. The media content for instruction and learning activities is imported, then linked according to various relationships. The author can add learner feedback to activities. Then the author designs teaching strategies so that the content and feedback are presented and sequenced more intelligently. The system is highly usable because most of the teaching strategy authoring is done by manipulating a set of sliders that specify preferences for such things amount of student control, position of testing, prefer specific/general material first, and amount of help provided. A second set sliders describe domain content using features such as familiarity with material, difficulty, general etc. Such sliders are set to categorize content according to these parameters, and again to define teaching strategies that have preferences for each parameter. The tutorial delivery program includes a "black box" rule-based expert system that uses all of these values to heuristically determine how to sequence content based on the learner's behavior over time, customizing content to what it thinks the user needs, according to the strategies developed by the author.

The hidden complexity problem becomes most evident when the author is test-running the tutorial and notices that it sequences some content or activity in an unexpected and unproductive way. Now it is the author's task to modify some of the strategy sliders (or some other setting) to "debug" the tutorial. But the system is complex enough so that the results of setting a slider to one value vs. another can not be easily predicted for the average tutorial session where there are many possible paths that students can take. In

other words, though the tool has made the task of authoring as easy as setting a few slider values, in order for the author to skillfully debug the inevitable imperfections in her initial design, she must have a sufficient mental model of the internal workings of the system to be able to predict the relationship between a particular state of the input variable (the authored settings) and the output behavior (the tutorial). This level of complexity is of course what we were trying to spare the author from in the first place by providing an easy to use tool. The situation is not always so dire. Redeem has been tested with many authors and Ainsworth has noticed that some authors build relatively simple tutorials that are easy to maintain and debug, while others do indeed create more complexity than they can skillfully manage.

Conclusions

Summary of main points. In this article I have explored the tradeoffs between generality/power and simplicity/usability in authoring tools for advanced and adaptive learning environments. I have emphasized the critical role of teachers and other subject matter experts in the design process. The need to have SMEs participate intimately in LE design acts as a forcing function toward authoring tool usability. As indicated in the discussion on "tradeoffs," no authoring tool is completely generic. Each one embodies certain assumptions and limitations regarding how domain knowledge, diagnostic strategies, teaching goals and strategies, etc. are represented. In a sense each one embodies a particular theory of knowledge and instruction. Though some are much more generic than others, generality comes at the price of simplicity or depth (for example common off the shelf multimedia authoring tools can be used to create shallow didactic tutors with multiple-choice interactivity for *any* subject area).

Lessons from activity theory. A human-computer interaction theory called activity theory, which draws much of its inspiration from Vygotsky's work, stresses how tools

and people evolve in a dialectic fashion. The theory originated with Vygotsky and colleagues, who analyzed human activity in terms of how it was mediated by designed artifacts (Vygotsky 1978, Kaptelinin et al 1995). Though people create tools, it is also the case that cognition, culture and society are dramatically affected by our tools. Authoring tools are not only tools that allow teachers and other SMEs to create learning environments, they challenge the author to reconceptualize instruction and content. In the research fields of learning technologies and AI expert systems there are pervasive anecdotal stories of how working with SMEs to capture their knowledge leads to reflection and analysis that results in the SME having a deeper or altered sense of their own subject area. This effect is even stronger with authoring tools, since they enable SMEs to participate deeply in design. But first they must assimilate several new mental models. They are encouraged to see their content in terms of modular units that can be flexibly ordered according to explicit situation-specific parameters. They are asked to articulate reasons for, and connection s between, the principles and concepts in their domain in new ways. They are tasked with articulating the implicit rules that they use to decide how to select and sequence topics, feedback, etc.

Authoring tools built for usability provide visual cues, working memory aides, and other types of scaffolding to help authors assimilate the appropriate representational structures, mental models, or underlying theories. Learning environments not only challenge teachers to reconceptualize their subject matter, but challenge them to reconceptualize how they see themselves as educators. Compared with traditional teaching methods, incorporating LEs into the classroom promotes a more learner-centered space. Students have more control over what they want to do or learn next. With each student or group engaged in its own process, the teacher must flexibly respond to issues and questions as they arise.

Tools designers as educational reformers. In conclusion, as we develop and deploy

authoring tools we should consider ourselves promoters of educational reform and transformation (in informal and industrial as well as formal education settings). Since advanced LEs are still rare in the "real world" of education and training, we are not so much helping teachers and educational developers more easily or cost effectively do something that they already do, rather, we are offering them the possibility of creating something entirely new, which can not be effectively used unless the whole enterprise of teaching is altered toward learner-centric approaches. Since authoring tool designers are, in a sense, exporting particular assumptions and models of knowledge and learning from their labs to practicing teachers (or content developers or SMEs) it behooves us to reflect deeply on exactly what assumptions and models are implicitly or explicitly contained in our systems.

References

Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B. & Wood, D. (2003). REDEEM: Simple Intelligent Tutoring Systems from Usable Tools. Chapter 8 in Murray, T., Blessing, S. & Ainsworth, S. (Eds.). *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht.

Collins, A. & Ferguson, W. (1993). Epistemic Forms and Epistemic Games: Structures and Strategies to Guide Inquiry. *Educational Psychologist*, 28(1), 25-42.

Halff, H, Hsieh, P., Wenzel, B., Chudanov, T., Dirnberger, M., Gibson, E. & Redfield, C. (2003). Requiem for a Development System: Reflections on Knowledge-Based, Generative Instruction, Chapter 2 in Murray, T., Blessing, S. & Ainsworth, S. (Eds.). *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht.

Kaptelinin, V., Kuutti, K., Bannon, L. (1995). Activity Theory: Basic Concepts and Applications. In B. Blumenthal et al. (Eds.) *Human-Computer Interaction. Lecture Notes in Computer Science*. Springer-Verlag, Berlin.

Merrill, M.D. (2003). Using Knowledge Objects to Design Instructional Learning Environments. Chapter 7 in Murray,

T., Blessing, S. & Ainsworth, S. (Eds.). *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht.

Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M, & Wogulis, J.L. (1997). Authoring simulation-centered tutors with RIDES. *International J. of Artificial Intelligence in Education*. Vol. 8 , No. 3-4, pp. 284-316.

Murray, T. (2003). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. Chapter 17 in Murray, T., Blessing, S. & Ainsworth, S. (Eds.). *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht.

Murray, T., Blessing, S. & Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments*. Kluwer Academic Publishers, Dordrecht.

Murray, T., Woolf, B. & Marshall, D. (2004). Lessons Learned from Authoring for Inquiry Learning: A tale of authoring tool evolution. To appear in proceedings of ITS-2004, Brazil, August, 2004.

Norman, D. (1988). *The Design of Everyday Things*. Doubleday, NY.

Shneiderman, B. (1998). *Designing the User Interface* (Third Edition). Addison-Wesley, Reading, MA, USA.

van Joolingen, W., & de Jong, T. (1996). Design and Implementation of Simulation Based Discovery Environments: The SMILSE Solution. *Jl. of Artificial Intelligence in Education* 7(3/4) p 253-276.

Vygotsky, L. S. (1978). *Mind in society: the development of higher psychological processes*. Cambridge: Harvard University Press.

Bio

Tom Murray is a research scientist working at the University of Massachusetts and Hampshire College. He has been researching and publishing in the area of advanced and adaptive learning environments since 1984. His projects related to authoring tools include Eon, MetaLinks, SimForest-G, and Rashi, described at http://www.tommurray.us.