Chapter 24 Invisible Intelligent Authoring Tools

Stephen B. Gilbert[1], Stephen B. Blessing[2]
[1] Iowa State University; [2] University of Tampa

Motivation

Imagine that you, an expert in technology and in the learning sciences, have decided to help your colleagues pass on their expertise to others by helping them build intelligent tutors systems (ITSs). Your expert colleagues can be in only one place at a time, and an ITS would multiply the impact of their expertise better than an online video, since an ITS can personalize the instruction. ITSs have demonstrated significant learning gains in a variety of disciplines, after all (Anderson, 1989; Koedinger, 1997; Lesgold, Lajoie, Bunzo & Eggan, 1992; Ritter, Kulikowich, Lei, McGuire & Morgan, 2007; VanLehn, et al., 2005), so this approach makes sense.

As you reflect on who these "expert colleagues" really are, you decide to focus on science, technology, engineering, and mathematics (STEM) topics, since the US has a dire need for more STEM expertise (Institute of Medicine, National Academy of Sciences & National Academy of Engineering, 2007), and since a wide variety of people have expertise that they would like to share with others, you'd like to focus on four kinds of experts: university faculty, K12 teachers, professional instructional designers and trainers, and high school students. Some of these experts will be reflective practitioners (Schön, 1983), who will bring a rich conceptual repertoire to the design of the tutor, while others will lack a conceptual model of the domain, much less the tutoring process. For this reason it is critical that the authoring tools be intelligent, i.e., able to adapt to the author.

You include the students as experts because you know that students can learn so effectively through the process of teaching and peer-mentoring (Biswas, Leelawong, Schwartz, Vye & The Teachable Agents Group at Vanderbilt, 2005; Crouch & Mazur, 2001), as well as from design-based activities (Kolodner, et al., 2003; Resnick, et al., 2009; Vattam & Kolodner, 2011). In fact, you realize, the process of formalizing knowledge into a particular representation can change the representation, so it would be worth studying all of your experts along the way, both to see if you can learn more about the basics of their conceptual change and to make sure that the tools you provide don't force undesired conceptual change on them. You know from Don Norman (1988) that a gap between your experts' expectations of your tools and their actual experiences with them will make the tools feel unnatural and frustrate your colleagues. Also, only the K12 teachers have actually received significant instruction on how to teach, so the authoring tools would have to incorporate good pedagogy implicitly, and the tutors that result from these tools would need to be evaluated for learning gains.

You then search the Internet for existing authoring tools for creating ITSs, and the results are disappointing. You find a summary of previous ITS authoring tools (Murray, Blessing & Ainsworth, 2003), but it offers more of a history than a guide to available tools. One chapter (Murray, 2003b) does offer lessons learned and guidance for the design of the ideal authoring tool; you'll keep that in mind. There are more recent search hits for ITS authoring tools, but most are academic papers, not software that's usable right now. Plus, some are all constrained to a specific domain, e.g., authoring algebra tutors. In terms of actual existing tutor authoring tools that you can sit down and use, six systems float to the top: Cognitive Tutor Authoring Tools (CTAT) (Aleven, Sewall, McLaren & Koedinger, 2006), the Extensible Problem-Solving Tutor (xPST) (Gilbert, Blessing & Kodavali, 2009), Authoring Software Platform for Intelligent Resources in Education (ASPIRE) (Mitrovic, et al., 2009), SimStudent (Matsuda, Cohen,

Sewall, Lacerda & Koedinger, 2007), ASSISTments (Razzaq, et al., 2009), and the Generalized Intelligent Framework for Tutoring (GIFT) (Sottilare, 2012).

As you examine these systems more carefully, however, you realize that none of them meet your desires entirely. The tools take different approaches to enabling the author to input and structure her knowledge. Some have a graphical user interface (GUI)-based system and some have what looks more like programming source code. Some might be easy for your experts to use to create simple problems, but require more computational thinking for more complex ones or for creating tutors that apply more generally. You find a comparison of the usability of CTAT and xPST (Devasani, Gilbert & Blessing, 2012), pointing out pros and cons to GUI vs. text-based approaches to tutor authoring, depending on the domain. That analysis suggests that the ideal ITS authoring tool, much like Adobe Dreamweaver or Microsoft Visual Studio, with both code views and views of the interface, should have multiple representations of content with which to interact. But the comparison article omits discussion of experts' individual differences. Surely two of your colleagues, even if they were experts in the same domain, might have different preferences of the kind of software they would like to use to represent their knowledge.

Delving more deeply into these authoring tools makes you realize the wide variety of interfaces for which ITSs have been built: equations and graphs, geometry and other diagrams, traditional desktop software applications, written text, virtual reality simulations for maintenance and repair, and Socratic dialogue, to name a few. You now realize that it would be best if there were a set of ITS authoring tools that could create tutors for all of these different interfaces, while catering to the individual differences of your colleagues and the needs of the particular knowledge domain. While that sounds like a significant challenge, you remember Don Norman's vision for the invisible computer (Norman, 1998), and his prescient early call for replacing a PC with many "information appliances" that would work for us while not burdening our minds or daily work. In effect, today's Internet cloud and mobile apps have begun to do that. App users seem happy to alternate the information representations with which they engage, when the representation feels natural for the chosen task. Therefore, you reason, the perfect ITS authoring tools would offer an invisible authoring system, one that allows your expert colleagues to, in effect, teach a computer what they know without getting in the way, just as naturally as a musician plays a composition into the iPad GarageBand app. Too bad those tools don't exist yet.

Introduction

A chapter in this volume by Blessing et al. offers a detailed comparison of the ITS authoring tools mentioned above. This chapter, in contrast, uses a user-experience lens to characterize the challenge of designing the ideally appropriate authoring tools to address the above scenario. Creating authoring tools that could meet this challenge would serve two purposes: (1) make it easier and faster to create ITSs much like the ones that exist today, and (2) create a framework that will lead to fundamentally better tutors in terms of pedagogy.

Definitions

To maintain clarity, we offer the following description of an intelligent tutor, its components, and the terms we are using. Shute (1994) notes that all ITSs contain the following: an expert model that contains the expert's knowledge; a student model that records what the student has learned (skills); and a pedagogical model that enables the tutor to react appropriately to the student's behavior. Other researchers note that the fourth critical component is the interface or problem-solving environment (Corbett, Koedinger & Anderson, 1997), which may be an off-the-shelf, third-party system or a

customized interface just for a particular tutor. To use standardized terminology proposed by Van Lehn (2006), the "task domain" is the discipline and content being taught, a "task" is a challenge assigned to a student or students, and tasks can be broken down into "steps." Finally, ITSs vary in the generality of their expert models. Some are example-tracing tutors, focusing on providing feedback around one specific example task. When we say "tutor," we intend the more general tutor, which contains knowledge that can be applied across many tasks of the same kind.

Extending the authoring task analysis described by Ritter, Blessing & Wheeler (2003), Table 1 contains the authoring tasks typically required to create a tutor. These steps are not strictly ordered; they typically are completed iteratively.

Table 2: Tasks of Authoring a Tutor

Characterize the Learning Environment	What are the possible actions or states of the learner's environment that need to be noted by the tutor? What objects will the learner manipulate, and what actions are permitted?
Organize the Curriculum	What topics and skills needed to be learned? Are there subskills?
Characterize the Learning Activities	What are the steps the learner needs to take, generally speaking? How do those steps demonstrate the skills that need to be learned, i.e., what is the mapping between steps taken and skills?
Describe Good Tutoring	What does a right answer look like? What are frequent wrong answers? How do you evaluate a learner's answer? What hints should be given if help is requested? What feedback should be given when the learner makes incorrect choices, and under what circumstances should it be given?

An Author's User Interfaces

The ideal user interfaces (UIs) that the expert will use to accomplish the above authoring tasks will likely vary between tasks and may vary by individual author, if the author's preferences for knowledge representation are known. Figure 8 illustrates example UIs that might be used.

The Wizard Dialog asks the expert a series of questions to refine the structure of the tutor, e.g., "Do your tasks have one right answer?" or "In your task domain, do students practice a task many times, 5–15 times, or fewer than 5 times?" The Wizard Dialog is essentially an expert system to narrow down to the most appropriate tutor structure.

The Decision Tree facilitates the creation of branching IF…THEN predicates. The idea of predicates and a predicate hierarchy is based on the original ACT-R inspired tutors, built using production rules (Anderson, Boyle, Corbett & Lewis, 1990), as well as on Carnegie Learning, Inc.'s SDK authoring tool (Blessing, Gilbert & Ritter, 2006; Blessing, Gilbert, Ourada & Ritter, 2009b).

The Click & Annotate UI works similarly to Camtasia or other software for making instructional screen-capture videos, in which the expert highlights elements of the interface and adds specific instructions. This UI would be particularly helpful for creating tutors on diagrams (e.g., free-body diagrams, blueprints, or geometry proofs) and with tutors on software applications.

The natural language scripting UI allows the expert to define a set of nouns (objects), adjectives (properties), and verbs (relationships), and then use natural language to describe conditions and interactions. In a game-based tutor, for example, an expert who is interested in guiding the student to stay close to the walls when exploring unknown corridors might write:

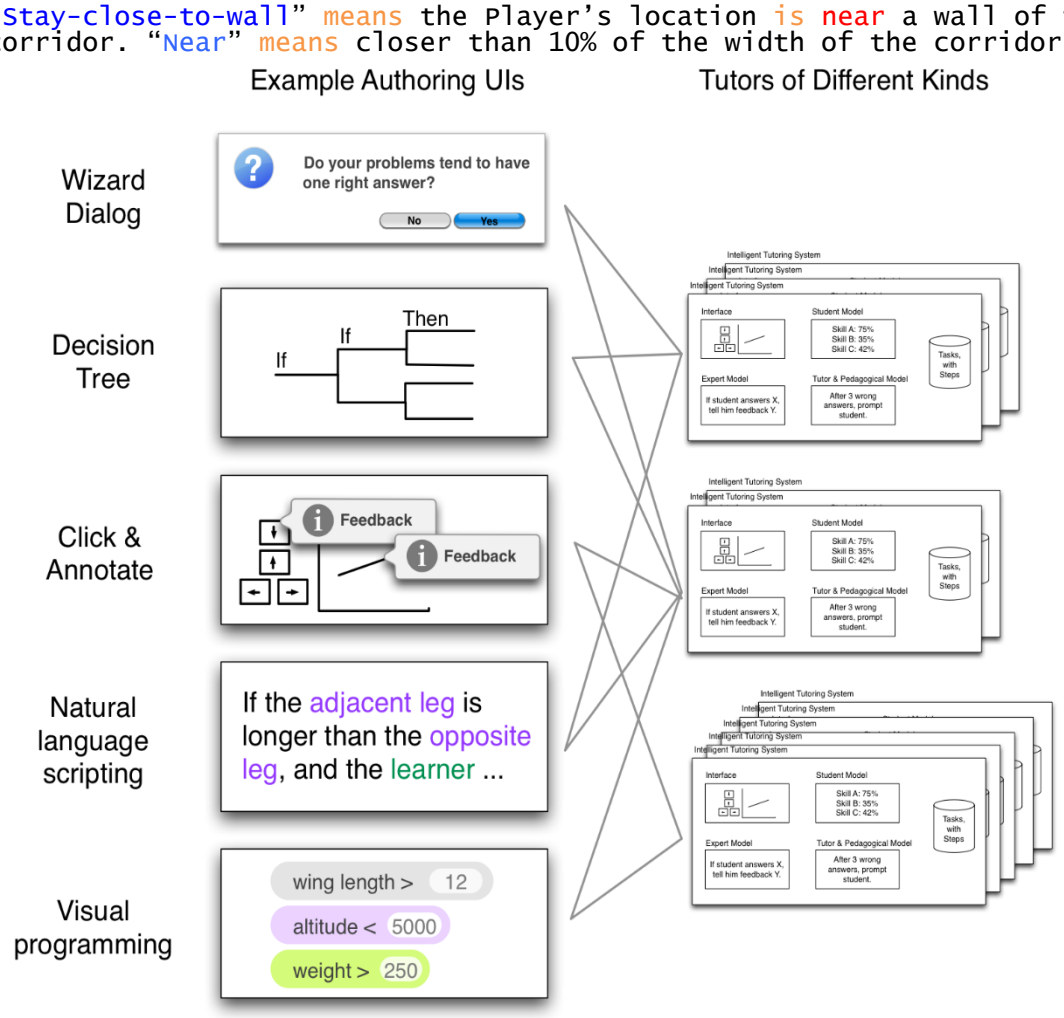

Figure 8: Different user interfaces are appropriate for different authoring tasks.

This approach uses principles of Applescript (2007) and Tutorscript (Blessing, et al., 2006), a language used with Cognitive Tutors at Carnegie Learning, Inc. A similar idea is proposed in the form of Natural-K (Jung & VanLehn, 2010).

The visual programming UI is similar to those used by Alice (Pausch, et al., 1995), Scratch (Resnick, et al., 2009), and Greenfoot (Henriksen & Kölling, 2004). The expert is given a set of primitives and operators and assembles them as blocks. This UI will be particularly useful for defining characteristics of the interface state, e.g., which components of a simulation are powered on, or the positions of entities within a serious game. One approach to tutoring based on game state is described in our previous work (Devasani, Gilbert, Shetty, Ramaswamy & Blessing, 2011; Gilbert, Devasani, Kodavali & Blessing, 2011).

This list of UIs is not exhaustive. Other UIs not depicted, for example, include a state transition graph, like the CTAT behavior graph (Aleven, et al., 2006), as well as a UI for constructing natural language parses and phrase classifiers, such as the Concept Grid (Blessing, Devasani & Gilbert, 2012; Devasani, Aist, Blessing & Gilbert, 2011). And of course plain source code or extensible markup language (XML)

is a possibility. Other new forms of UIs will become appropriate as new kinds for tutors arise, e.g., interfaces to allow conditions based on a student's affect or motivation.

A tutor would likely be created using a combination of these UIs. A *tutor template* would be a particular configuration of UIs designed to help create a tutor of a specific kind. As mentioned above, Norman's information appliances are the inspiration, so that the ideal ITS authoring system becomes an invisible collection of tools well designed for their purposes. Just as mobile phone users are familiar with switching between an email app, a calendar app, and a contacts app, which all share data, tutor templates will provide an optimal configuration of UIs for a given task domain, and that the backend will allow appropriate data sharing across them. In addition, a Template Recommender could be created, an expert system that will recommend templates to tutor authors based on a Wizard-style interview about the discipline and learning goals.

GIFTscript

While natural language scripting is mentioned above, it is worth considering it in more detail because of the power of allowing an expert to use her own language to create a tutor. Statements like the above "stay-close-to-wall" include the definition of new concept (stay-close-to-wall) and of a new condition ("near"). It assumes that the objects "corridor" and "wall" have been defined elsewhere and that a property "width" is associated with "corridor" (perhaps inherited from a parent object such as "object"). The approach suggests an object hierarchy with properties assigned to the objects that could be considered adjectives, e.g., a generic "building" object might have properties of "height," "location," and "dimensions" that could be inherited by child objects "house," "wall," "store," etc. Those child objects could in turn have specialized properties. This approach is taken by Carnegie Learning's tutor authoring tools (Blessing, et al., 2006), but user interfaces don't take advantage of natural language to author these hierarchies on the fly. And, working with inheritance hierarchies requires a degree of computational thinking (Wing, 2008) that our target users probably do not all have.

We suggest that if users could think of their knowledge domain in terms of scenarios, tasks, and concepts (or perhaps skills), an interactive language-based authoring tool might be able to be created, which creates the aforementioned object hierarchy more naturally. Since there is interest in developing the perfect authoring tools for GIFT, we call this language GIFTscript.

GIFTscript would be integrated with the other UIs described above by using text editing boxes that check GIFTscript syntax. Special visual indicators on the boxes would indicate to the user that (1) GIFTscript is available, (2) that syntax is violated, and (3) that valid GIFTscript is present. Also, these text boxes will support auto-completion and color coding, much like Visual Studio. A mockup of a sample interface is provided in Figure 2 as illustration. In this example, using some of the primitives already extant in GIFT, you could imagine script such as

```
"Avoid Location {x}" means player location is far from {x} location.
```

The interface would recognize that the user had defined a new condition called Avoid Location. It would also recognize noun phrases (objects) such as "player location" and "location," and if it didn't recognize those noun phrases, it would ask the user to define them. In this example, it recognizes a new adjectival phrase, "far from," and asks for a definition. The user might use some known objects to define it:

```
"far from" means distance > than 20 ft.
```

Then, using the new Avoid Location condition, the author could write rules such as

```
If the Player does not avoid location enemy bunker then remind, “Stay a
safe distance from the enemy.”
```

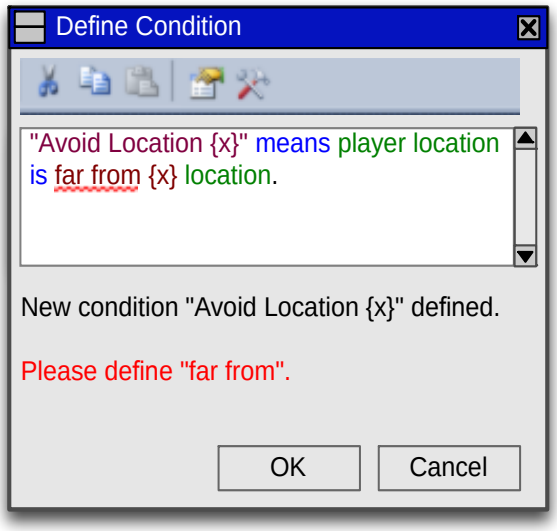

Figure 9: Mockup of interactive dialogue box for editing GIFTscript.

The critical feature of a UI featuring GIFTscript is offering a usable visualization of the objects and structures resulting from this approach.

Authoring Tool as Research Tool

Just as many creators of learning technologies are studying the misconceptions of their learners and getting usability feedback via educational data mining techniques and the use of embedded or stealth assessment (Shute & Ventura, 2013; Shute, Ventura, Bauer & Zapata-Rivera, 2009), it will be valuable if ITS authoring tools are themselves instrumented with click-stream logging similar embedded assessments. Using these methods, authors’ own conceptual change can be monitored, especially if combined with a pre- and post-assessment of the author’s understanding of the domain. Even without such assessments, however, an unsupervised learning classification method could be used to cluster different authors as to their approaches to knowledge representation, and perhaps, the interface elements could be personalized to each author’s style. Also, it has been found useful in previous studies of authoring tools, e.g., Blessing, Gilbert, Ourada, and Ritter (2009a) to monitor the time spent by the author in different components of the authoring system. These usage profiles can be used to broadly characterize authors as experts or novices and perhaps give intelligent tutoring-style feedback to the authors themselves on the task of authoring.

Recommendations and Future Research

To create a mapping between types of tutors and appropriate UIs for authoring, it will be important to create a taxonomy of tutor types. The larger categories of tutor interfaces might be, for example, diagrams, equations, text, procedures, and cases. Each larger category might have subcategories, e.g., text-based tutors categorized by focus on short-answer responses, longer text passages, English language learning, reading, or Socratic dialogue. While other researchers have provided overviews of the gamut of intelligent tutors (Murray, 2003a; Sottilare, 2012; VanLehn , 2006), they have not focused on categorizing

tutors by the kind of authoring approach required, or by the knowledge representation required of an expert author. A taxonomy effort such as this might lead to a table as shown in Figure 3, which could then guide the creation of the ideal invisible authoring tools, for GIFT or other tutoring systems.

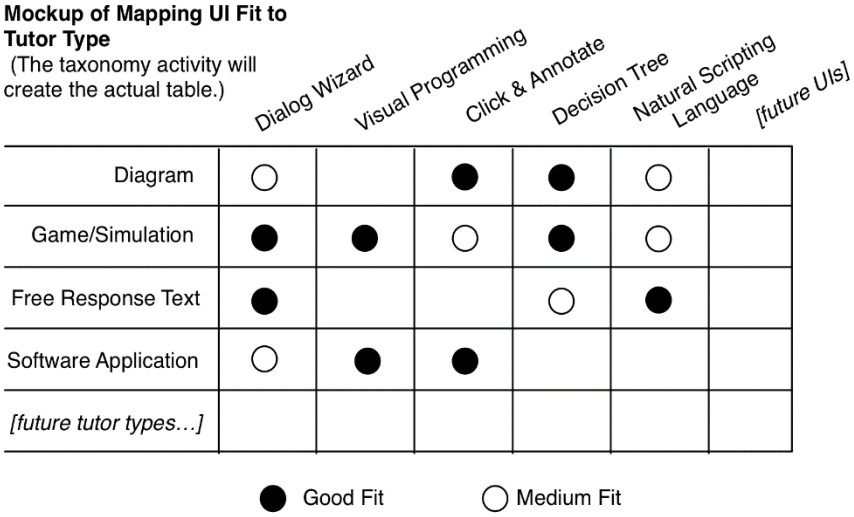

Figure 10: A mockup of the type of matrix that might emerge from mapping a taxonomy of tutor types to user interfaces for authoring them.

Research questions remain that are worth exploring. Are there noteworthy individual differences in UI preferences among experts in the same field? How does authoring with a given UI affect an expert's own understanding of his expertise? What effect does the choice of a tutor's authoring UI have on student learning? Does having natural UIs for authoring improve the quality of the resulting tutor?

References

Apple (2007). Introduction to AppleScript Overview Retrieved July 6, 2012, from http://developer.apple.com/applescript/

Aleven, V., Sewall, J., McLaren, B. M. & Koedinger, K. R. (2006). *Rapid Authoring of Intelligent Tutors for Real-World and Experimental Use.* Paper presented at the Sixth International Conference on Advanced Learning Technologies.

Anderson, J. R., Boyle, C. F., Corbett, A. T. & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial intelligence, 42*(1), 7-49.

Anderson, J. R., Conrad, F. G. & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*, 467–505.

Biswas, G., Leelawong, K., Schwartz, D., Vye, N. & The Teachable Agents Group at Vanderbilt. (2005). Learning by Teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence, 19*(3-4), 363-392. doi: 10.1080/08839510590910200

Blessing, S., Gilbert, S. B. & Ritter, S. (2006). *Developing an authoring system for cognitive models within commercial-quality ITSs.* Paper presented at the Nineteenth International FLAIRS Conference.

Blessing, S. B., Devasani, S. & Gilbert, S. B. (2012). *Evaluating ConceptGrid: An Authoring System for Natural Language Responses.* Paper presented at the Twenty-Fifth International FLAIRS Conference, Marco Island, FL, USA.

Blessing, S. B., Gilbert, S. B., Ourada, S. & Ritter, S. (2009a). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education, 19*(2).

Blessing, S. B., Gilbert, S. B., Ourada, S. & Ritter, S. (2009b). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education*.

Corbett, A. T., Koedinger, K. R. & Anderson, J. R. (1997). Chapter 37 - Intelligent Tutoring Systems. In G. H. Marting, K. L. Thomas & V. P. Prasad (Eds.), *Handbook of Human-Computer Interaction (Second Edition)* (pp. 849-874). Amsterdam: North-Holland.

Crouch, C. H. & Mazur, E. (2001). Peer Instruction: Ten years of experience and results. *American Journal of Physics, 69*(9), 970-977.

Devasani, S., Aist, G., Blessing, S. & Gilbert, S. B. (2011). *Lattice-Based Approach to Building Templates for Natural Language Understanding in Intelligent Tutoring Systems.* Paper presented at the Proceedings of the Fifteenth Conference on Artificial Intelligence in Education, Auckland.

Devasani, S., Gilbert, S. B. & Blessing, S. B. (2012). *Evaluation of Two Intelligent Tutoring System Authoring Tool Paradigms: Graphical User Interface-Based and Text-Based*. Paper presented at the Twenty-First Conference on Behavior Representation in Modeling and Simulation (BRIMS), Amelia Island, FL, USA.

Devasani, S., Gilbert, S. B., Shetty, S., Ramaswamy, N. & Blessing, S. (2011). *Authoring Intelligent Tutoring Systems for 3D Game Environments.* Paper presented at the Proceedings of the Authoring Simulation and Game-based Intelligent Tutoring Workshop at the Fifteenth Conference on Artificial Intelligence in Education, Auckland.

Gilbert, S. B., Blessing, S. B. & Kodavali, S. (2009). *The Extensible Problem-Specific Tutor (xPST): Evaluation of an API for Tutoring on Existing Interfaces.* Paper presented at the 14th International Conference on Artificial Intelligence in Education.

Gilbert, S. B., Devasani, S., Kodavali, S. & Blessing, S. (2011). *Easy Authoring of Intelligent Tutoring Systems for Synthetic Environments*. Paper presented at the Twentieth Conference on Behavior Representation in Modeling and Simulation (BRIMS), Sundance, UT, USA.

Henriksen, P. & Kölling, M. (2004). *Greenfoot: Combining object visualisation with interaction.* Paper presented at the Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications.

Institute of Medicine, National Academy of Sciences & National Academy of Engineering. (2007). *Rising above the gathering storm: Energizing and employing America for a brighter economic future*. Washington, D.C.: National Academies Press.

Jung, S.-Y. & VanLehn, K. (2010). *Developing an intelligent tutoring system using natural language for knowledge representation*. Paper presented at the Intelligent Tutoring Systems Conference (ITS 2010).

Koedinger, K. R., Anderson, J.R., Hadley, W.H. & Mark, M.A. (1997). Intelligent tutoring goes to school in the big city. *International Journal for Artificial Intelligence in Education, 8*, 30-43.

Kolodner, J. L., Camp, P. J., Crismond, D., Fasse, B., Gray, J., Holbrook, J., et al. (2003). Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design (tm) into practice. *The journal of the learning sciences, 12*(4), 495-547.

Lesgold, A., Lajoie, S., Bunzo, M. & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin & R. Chabay (Eds.), *Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G. & Koedinger, K. R. (2007). Predicting students' performance with simstudent: Learning cognitive skills from observation. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS, 158*, 467.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., et al. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education, 19*(2), 155-188.

Murray, T. (2003a). An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art *Authoring tools for advanced technology learning environments* (pp. 491-544): Springer.

Murray, T. (2003b). Principles for pedagogy-oriented knowledge based tutor authoring systems: Lessons learned and a design meta-model *Authoring Tools for Advanced Technology Learning Environments* (pp. 439-466): Springer.

Murray, T., Blessing, S. & Ainsworth, S. (Eds.). (2003). *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-effective Adaptive, Interactive, and Intelligent Educational Software*. Norwell, MA: Kluwer Academic Publishers.

Norman, D. (1988). *The design of everyday things*. New York: Basic Books.

Norman, D. (1998). *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. Cambridge, MA: MIT Press.

Pausch, R., Burnette, T., Capeheart, A., Conway, M., Cosgrove, D., DeLine, R., et al. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications, 15*(3), 8-11.

Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., et al. (2009). The Assistment Builder: Supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on, 2*(2), 157-166.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: programming for all. *Commun. ACM, 52*(11), 60-67. doi: http://doi.acm.org/10.1145/1592761.1592779

Ritter, S., Blessing, S. B. & Wheeler, L. (2003). Authoring tools for component-based learning environments. In T. Murray, S. Blessing & S. Ainsworth (Eds.), *Authoring Tools for Advanced Technology Learning Environments* (pp. 467-489). Norwell, MA: Kluwer Academic Publishers.

Ritter, S., Kulikowich, J., Lei, P., McGuire, C. L. & Morgan, P. (2007). What evidence matters? A randomized field trial of Cognitive Tutor Algebra I. In T. Hirashima, U. Hoppe & S. S. Young (Eds.), *Supporting Learning Flow through Integrative Technologies* (Vol. 162, pp. 13-20). Amsterdam: IOS Press.

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York: Basic books.

Shute, V. & Ventura, M. (2013). *Stealth assessment: Measuring and supporting learning in video games*: MIT Press.

Shute, V. J. & Psotka, J. (1994). Intelligent tutoring systems: Past, present, future. *Technical Report AL/HR-TP-1994-0005, USAF, Armstrong Laboratory*.

Shute, V. J., Ventura, M., Bauer, M. & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning. *Serious games: Mechanisms and effects*, 295-321.

Sottilare, R. (2012). *Considerations in the Development of an Ontology for a Generalized Intelligent Framework for Tutoring* Paper presented at the International Defense and Homeland Security Simulation Workshop 2012, Vienna, Austria.

VanLehn , K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education, 16*(3), 227-265.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). *The Andes Physics Tutoring System: Five Years of Evaluations*. Paper presented at the Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology.

Vattam, S. S. & Kolodner, J. L. (2011). On foundations of technological support for addressing challenges facing design-based science learning. *Technology Enhanced Learning and Cognition, 27*, 233.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society, 366*, 3717-3725.