A more revolutionary role for educational software is possible.

How Should Students Learn?

Andrea A. diSessa
Graduate School of Education
University of California, Berkeley
disessa@soe.berkeley.edu
http://www.soe.berkeley.edu/boxer.html

Mark Guzdial makes an important point, one with which I wholeheartedly agree. Software intended for learners must be designed in different ways from the vast majority of pieces of software we see in wide distribution. Even more, Guzdial and I agree on many values and strategies for producing productive computer-supported learning environments. Perhaps most important, we both would like to see students learning in modes that provide them wide flexibility and freedom for creativity in undertaking tasks they feel are personally meaningful, while at the same time learning important subject matter. Accomplishing and learning, together, is a goal that can make learning more effective and more engaging; it is a goal that software designed with narrow instrumental goals—such as merely accomplishing a particular task—will almost always fail to achieve.

Nonetheless, there are some differences in our points of view that I will try to expose in this brief essay. These differences are partially strategic—how should one best achieve agreed ends?—and partially teleological—what should we strive to achieve? More than anything else, I believe our differences stem from a slightly different take on the "big picture" in which we need always to situate our local design decisions. Given how much we agree on, especially in view of how close together we stand in the wider universe of approaches to computer-supported learning, my points may seem at first to pick nits. Nonetheless, bringing out differences will allow me to make points I feel are important.

The first time I read through Guzdial's paper, I felt a vague unease at how he portrayed learners. He sees learners as perhaps overwhelmed by a "huge hierarchy of goals," somewhat bored—or at least in need of "enormous motivation" to work through these goals—and in dire need of help, for example, in terms of scaffolding that can keep their heads above water. Of course, I'm exaggerating for effect. But, still, students come across as a little lost and helpless in his portrayal. It is certainly important to have empathy for students, and there can be no question our job is to help them. But, I don't believe we should assume that these unfortunate conditions are the permanent fate of students. Instead, I think it is extremely productive to take as one of our central goals to eradicate the conditions he describes as affecting students, rather than accepting them and trying to deal with them.

I propose that our long term goals for learning environments should include helping students to become *committed learners* (diSessa, in press). More than anything else, I would characterize committed learning as engaging in learning tasks that feel important and personally fulfilling, and engaging in those tasks with a sense of confident competence. Most of my life involves extended and felt-to-be-meaningful activities. We should have no less a goal for our students.

Committed learning may be difficult to achieve. But that doesn't mean we shouldn't try for it. Indeed, treating students as in need of motivation and without sufficient resources for the tasks we pose to them is undoubtedly a self-fulfilling prophesy. Always trying to "motivate" our students without cultivating their own independent motivation, and assuming the tasks we propose will always be ones student would not chose for themselves gives away too much of what we might achieve, and it lets us off the hook too easily. In this state, we'll endlessly patch the system, without looking for the fundamental overhaul it really needs.

I believe we are too used to "school as usual." That "tremendous hierarchy of goals" is really our problem, not the learners. As Guzdial remarked at the end of his paper, experts are probably (I would say certainly) not so different from learners-as-users. I have many, many goals in producing this paper. I want to extend Guzdial's points without undermining his basic presumptions. I want to make some space for somewhat different design heuristics. I would like to speak eloquently for attempting deep educational reform. And, I have many low-level goals too. For example, generating good sentence and paragraph structure is never a trivial accomplishment—at least for me. All the while, I expect to be learning, too: learning to write; learning about my own position on technology use in education; learning about Guzdial's accomplishments by thinking about them. The big difference between me and students, as portrayed in Guzdial's article, is that my plethora of goals feels natural because I am committed to them, and I feel competent to engage them.

Committed learning may seem pie in the sky for many readers. On the other hand, most if not all of the works cited by Guzdial in his paper show examples of committed learning as much as they show examples of software strategies or the difficult position we put our students in when we accept "school as usual" as a necessary condition of learning. Again and again, if not all the time, my own project has managed to create conditions for committed learning. I refer the reader to diSessa (in press) for an extensive presentation.

At this point, I want to shift attention to software, although committed learning will return quickly. I want to introduce a competing model of software to that which Guzdial showed. Although he mentioned Boxer (http://www.soe.berkeley.edu/boxer.html) approvingly (thank you), he missed some of the important points I would make about it. Boxer is a system designed for a "literacy model" of the use of technology in education. That is, we are experimenting with the goal of having every participant in the educational process, especially students and teachers, become fluent over many years with a computational medium. A computational medium, of which Boxer is a prototype, enfolds written text, but extends that traditional literacy basis with new forms of organization (like hierarchically nested boxes, hyperlinks, and so on) and, more importantly, it includes the capacity to create dynamic and interactive representations, not just static ones. (The latter capacity most people call "programming.") Our central hypothesis is that a computationally enhanced medium can't fail over the long term to provide civilization with huge additional intellectual advantages on the scale of what plain textual literacy has done for us.

The literacy model has students learning one very rich piece of software, a computational medium, and reusing that skill again and again over many years in multiple contexts.

Let me talk about how the literacy model intersects and differs from the picture of "good software" that Guzdial presented. First, a simple point: A lot of the difficulties that Guzdial describes are not the result of an unavoidable "huge hierarchy of goals," but simply that students need to learn a piece of software at the same time as they engage in other, important learning activities. The literacy model has students learning one very rich piece of software, a computational medium, and reusing that skill again and again over many years in multiple contexts. This is just like written literacy works now. We don't learn a new "pidgin" language anew for each school subject. Instead, we learn English reading and writing, and we use it while at the same time learning more about it in every course we take. To make the point concrete, Guzdial's students would be tremendously better off if they had already learned a very flexible and programmable computer system, and then, on that basis, learned the "language independent" strategies he teaches in his course. Our students start learning a respectable amount of programming at least as early as sixth grade.

The larger point is that if we have inflicted a panoply of goals from which students swoon, we have abrogated our responsibility in instructional design. Our responsibility includes, for example, to spread learning out over appropriately long time-scales, and to make sure each activity is continuous with the skill set with which students come into it. From the perspective of committed learning, we have one other obligation as instructional designers. We need to make sure the activities are also continuous with the goals and interests of our students. If we have constantly to "motivate," we're too late.

Guzdial's use of Smalltalk is poignant. Smalltalk's original goals were exactly the literacy model, to become a generic computational medium for students and teachers (Kay and Goldberg, 1977). But a recognizable dynamic of falling back into old ways, and aiming for "users," not "learners-as-users" as Guzdial puts it, took over. Smalltalk became an environment for professional programmers. The case of Smalltalk makes the main point on which Guzdial and I agree. Too much software is designed to suit narrow needs, and not those of learners. But it also makes my point about looking to the broader picture. Guzdial should not have to make do, patch, and supplement software written for nonlearners.

Guzdial notes that Boxer does not have a lot of specialized structures like the ones he designs into his environments. This is true, but it is true in the same way for the natural language, the basis of standard literacy. Although children speak variations of the language, there is no special "English with training wheels" to support learners. (Not that there is anything wrong with training wheels. But training wheels are always transient in the longer course of learning and use. More importantly, we should never let the possibility of training wheels excuse a poorly designed bike.) The complementary strategies that *are* quite evident in standard literacy are important to recognize. To be very brief, we depend on social and material cumulativity. That is, over a long time of creating a literate culture, we develop a substantial literature that is adapted to different foci and levels of learning (including children's books), and we develop wide-spread know how about how to introduce children to the life-long task of becoming literate.

My project can't have created any true wide-spread literacy in the small experiments we have done with Boxer. But still, social and material cumulativity is evident and manifestly part of why we've managed to succeed (sometimes) in creating environments for committed learning. For example, our classes are full of tutorials and reference materials generated in prior versions. Even in the earliest Boxer class, students, as well as teachers, contributed to cumulativity. For example, a sixth-grade student wrote the first programming tutorial I recall in Boxer. His teacher picked that up, modified it. and used it with more classes. The tutorial I now use and distribute with Boxer is a modification of that same Boxer document! (For a more extended treatment of this event, consult diSessa (in press).) In terms of nonmaterial cumulativity, we still use some of the same activities that proved so successful in the first class (diSessa et al., 1991).

One of the other wonderful occurrences in our first Boxer class was that the teacher instigated a Boxer "library" into which students put their creations. Like Guzdial's CoWeb, our children's personal motivations and accomplishments could feed back into the community in many ways. For one, many students at the school learned Boxer by coming to play with other students who had made interesting things in Boxer and put them into the library. The Bentley Boxer library illustrates many of the principles of collaboration and students helping students that Guzdial showed us. But it also illustrates my more general point about social and material cumulativity. In addition, the Boxer library worked because Boxer is general and flexible enough to encompass many forms, including communal libraries, without the need for "specialized software" or "learning adaptations of standard software." (At the time, the only structure that was needed for the library was networked file-serving. Now, Boxer has additional generic structures that can be used for such communal creations. For example, we have a completely integrated mail system that allows users to send any Boxer structures to each other. Files are first class Boxer objects that can be integrated in any larger Boxer structure, such as a library or database. And we have "net boxes" that allow students to share boxes transparently over the Internet. Somewhat like links on a web page, net boxes expand and fill with the remote Boxer material when clicked on.) Some explanation of why "specialized software" is not necessarily needed to support collaboration is provided in diSessa (1995a).

As wonderful as the WikiWikiWeb and Guzdial's use of it are, I think these once again constitute a case of overcoming traditions of software design that would best be left in the past. If the Web is to be anything like a computational medium, how could the idea that people should easily edit an existing thing have come as an afterthought? In Boxer, we've designed for writing as well as reading. Every user of Boxer gets used to modifying and changing anything he or she encounters as a natural part of the dynamic of use. The naturalness of modifying and extending things created in Boxer has become so much a part of the way we do things that, these days, I almost always think of what I produce as "tool kits" that I expect teachers and learners (and other developers) to change and extend (diSessa, 1997). One of our most recent successes was to produce a tool kit for scientific visualization (Friedman and diSessa, in press). A small part of our group provided resources for the rest (the core tool kit), and the rest of the group very quickly developed a substantial range of materials for students to learn with. The coup de grace is that students, themselves, modified and extended the toolkit and materials to serve their own ends.

To escalate the point, while Guzdial sees STABLE and CoWeb as examples of the same design heuristics, I see them as telling very different stories. STABLE is an example of patching or extending old software in very specialized ways, and so as to preserve the hegemony over goals and means that school tends to perpetuate. I view CoWeb as a really different experiment of putting generic computational power in the hands of students and instructors, and letting them work out, over an extended time, what can and should be done with it. *Design for and count on both social and material cumulativity in an evolving literate culture.* The fact that students took off with "off task" creations (Guzdial calls them "non-traditional") like songs, games and essays, but also with many innovative and helpful "on task" creations is a sure sign that committed learning took hold. Students felt that their own goals could be served by this software, and, sure enough, many (but not all) of those goals included goals the class instructor would very much approve of. Fundamentally, the domain of action was not restricted by an overly presumptuous piece of software that acted as if it knew what was right for students and even for future teachers of the class. In the CoWeb experience, students were not incompetent and in need of tremendous support. Instead, they were empowered by software that was as easy to use in production mode (writing) as well as in consumption (reading). The students were not overwhelmed with a hierarchy of goals because, in many instances, they owned the agenda. If there was a huge hierarchy, it was *their* huge hierarchy.

Guzdial emphasizes, sometimes subtly, sometimes more explicitly, that a lot of his attempts are to make do in the context of a system of learning and premade software he did not design. Certainly realism is an important attribute in attacking educational issues. In contrast, I've emphasized the importance of thinking about the bigger picture. In the face of his realism, some of my considerations may seem like naive revolutionary rhetoric. But at least we should be clear what we'd like in the best possible world, and compromise from there. In *my* best possible world, students are committed learners and are not people we have to feel sorry for, constantly motivate and support if they are ever to get anything done. Instead, we cultivate competence over an extended period of time, and we can count on social and material cumulativity, rather than on patches and narrowly goal-directed pieces of software to make things work. Guzdial's successes, and those he cites (including those of our project), seem to suggest that we may not need to compromise so much as we might have thought. Given a chance, committed learning can take hold. In terms of convincing a broader community that flexible, learner-centered software is important, I'm standing with Guzdial. I am ready to announce that we need a revolution, even if I don't expect it to happen right away.

References

diSessa, A. A. (1995a). Collaborating via Boxer. In L. Burton and B. Jaworski (Eds.), *Technology—A Bridge between Teaching and Learning Mathematics*. (pp. 69-94). Bromley, Kent, UK: Chartwell-Bratt.

diSessa, A. A. (1995b). The many faces of a computational medium. In A. diSessa, C. Hoyles, R. Noss, with L. Edwards (Eds.), *Computers and Exploratory Learning*. (pp. 337-359). Berlin: Springer Verlag.

diSessa, A. A. (1997). Open toolsets: New ends and new means in learning mathematics and science with computers. In E. Pehkonen (Ed.), *Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, Vol. 1. Lahti, Finland, 47-62. (ftp://soe.berkeley.edu/pub/boxer/Distribution/toolsets.pdf).

diSessa. A. A. (in press). *Changing minds: Computers, learning and literacy*. Cambridge, MA: MIT Press.

diSessa, A. A. , Hammer, D., Sherin, B. and Kolpakowski, T. (1991). Inventing graphing: Meta-representational expertise in children. *Journal of Mathematical Behavior*, *10*(2), 117-160.

Friedman, J. and diSessa, A. A. (in press). What should students know about technology: The case of scientific visualization. *Journal of Science Education and Technology*.

Kay, A., and Goldberg, A. (1977). Personal Dynamic Media. *IEEE Computer*, *10*(3), 31-42.