Interactive Learning Environments
2003, Vol. 00, No. 0, pp. 000–000

Reflections on Component Computing from the Boxer Project's Perspective

Andrea A. diSessa
Berkeley Boxer Project, Graduate School of Education, University of California, Berkeley, CA, USA

ABSTRACT

The Boxer Project conducted the research that led to the synthetic review ‘‘Issues in Component Computing.’’ This brief essay provides a platform from which to develop our general perspective on educational computing and how it relates to components. The two most important lines of our thinking are (1) the goal to open technology's creative possibilities to the broadest range of people possible, and (2) to do so on the basis of a single architecture, a computational medium, that includes the possibility of constructing and modifying interactive, dynamic constructions by all participants.

INTRODUCTION

This note situates the Boxer Project's thinking about component computing within the perspective of our larger body of work. In ‘‘Issues in Component Computing’’ we took pains to present a balanced view, based on data. Here, we make sure our broader agenda and predispositions are not hidden. Readers can then make their own judgments about our objectivity in analysis. In addition, in ‘‘Issues in Component Computing’’ we attempted to keep separate analytically distinct threads (e.g., as represented in our use of models as *partial* specifications); this is a chance to weave threads together in a way that makes best sense, given our general philosophy of educational computing.

Democratization

The Boxer Project's general approach to educational computing has been, first and foremost, to attempt to make the technology itself as open and broadly

Address correspondence to: Prof. A. diSessa, University of California at Berkeley, Graduate School of Education, EMST, 4533 Tolman Hall #1670, Berkeley, CA 94720-1670, USA. E-mail: disessa@dewey.soe.berkeley.edu

useful as possible. We feel it is an unacceptable compromise to design systems that, a priori, exclude teachers and students from direct, creative contributions. In ‘‘Issues in Component Computing’’ we discussed openness mainly for teachers in order to keep the report brief and to focus on areas common across component projects. But, in other places we have systematically tried to explain and document the value of open technologies for students (diSessa, 2000).

Our position is not that we know how much teachers and students can gain from open environments; that is still at least somewhat uncertain.[1] Instead, we believe it is unacceptable to *presume* – in the face of the extended and just-begun period of social adjustment to the affordances of technology – that shutting out non-technology expert contributions is the right way to go. The presumption that ‘‘one-way’’ media (experts make things that users merely use) are best for education is almost certainly a self-fulfilling prophecy. Empowering teachers with open technology is not an established fact so much as a challenge that is both plausibly met and also well-worth taking on. It is a challenge that has at least encouraging existence proofs, and one that could, if we avoid it, threaten the best that can be achieved.

Cumulative Expertise

Instant competence with ‘‘transparent’’ technology is an illusion and dangerously restrictive assumption. Instead, just like reading and writing, competence with technology will prove its value as an extended accomplishment. These are, of course, very broad and deep issues on which the current state of the art has only a small bearing. For example, experiments with technology-naïve teachers just begin to set the grounds for adequate assessment of how much value, ultimately, teachers can gain from open technologies. We do not take high-levels of technological expertise as sensible prerequisite for engagement with computer-based learning. But neither is it sensible to presume high levels will never be accomplished, at least in a particular minority of teachers who might contribute creatively to the educational software pool, or show the way for other teachers.

[1]To be fair to our own work, we feel we know that students can gain an enormous amount from the use technology that accepts their creative input. For teachers, data is less clear.

THE LADDER MODEL AND OTHER MODES OF DEVELOPMENT

The LaDDER model has many attractive properties, especially within our larger framework of democratizing technology and empowering teachers. In particular, the model is particularly well-adapted to grass-roots development by teachers, and, more broadly, it is a good representative of methods to attempt to create autonomy among teachers. Teachers involved with LaDDER collaborations build their own capability to work independently, as well as contributing to a product. In contrast, integration teams predicated on ‘‘experts only’’ technology don’t obviously build autonomy or an intimate understanding of the specific affordances of technology for learning. Member-sustained communities may not provide enough support even to bring teachers seriously into the design enterprise.

The LaDDER model would seem also to soften some of the problems we discussed concerning discoordination across community, such as developers’ need to continue to develop without waiting for educationalists to ‘‘catch up.’’ In particular, teachers and local developers can do a lot of work without primary developers. Furthermore, educational design may always work best with iterative refinement over extended periods; the LaDDER model supports this without primary developers’ being constantly on call.

The toolset idea is integral to the LaDDER model: building tools that are, at the same time, flexible, yet also adapted to some particular area (e.g., to a curricular area). In addition, gradual development of toolsets and curricula supports the co-development of communities and technological resources, which has much to recommend it in view of the growing recognition that social structures are integral to any technological accomplishment (Bijker & Law, 2000).

Obviously the LaDDER model does not do all things well. We do not think it works well in early stages of designing infrastructural technology, such as Boxer, E-Slate, or the component infrastructure of ESCOT. Even the development of a family of compatible toolsets (perhaps built on a common core set of components) may work well *in conjunction with* LaDDER instances, but almost certainly such development needs other organizational supports, such as might be provided by a two-legged model. Similarly, we believe that procedural programming provides better conceptual support than wiring for certain topics at least (diSessa, 2000, chapter 2). The E-Slate project, its commitment to wiring for component interconnection notwithstanding, shares this perspective.

Although we emphasized the LaDDER model and toolsets here, our broader approach is much more eclectic. See diSessa (2000).

Technology Specifics

Programming plays a special role in our general point of view in several ways. First, we consider it an independently attractive activity for at least some curricular areas (e.g., motion and geometry). Second, in terms of component interconnect, we feel wiring is too specialized to be the ‘‘right’’ interconnect method in all instances. In ‘‘Issues in Component Computing’’ we noted that we chose to have images in our image processing toolkit automatically connect to ‘‘nearby’’ graphs. Programming affords a wide variety of interconnect protocols (see Parnafes & diSessa, 2001); primary developers may want to fabricate special-purpose interconnect protocols for certain tools or toolsets.

In addition, programming constitutes a significant way of adapting components to particular circumstances – not only in the construction of a toolset, but also in adapting individual tools or constructions using tools for teachers’ quite specific purposes. Since, as we argued, adaptability of individual components is much more important than is generally recognized, making it broadly accessible socially seems the best default strategy.

Boxer uniquely aims to have a common form for all loci of programming. We have a rich container that entails the capacities: (1) to use already-constructed software; (2) to do such minor things as add or edit text; (3) to copy, connect or re-connect components; (4) to open components for inspection or modification; and (5) to program for direct pedagogical purposes or to construct a component or a learning environment from scratch.

Having a single locus for the accumulation of technical competence maximally leverages any technical competence that users develop. Learning about Boxer structures and programming in any capacity transfers to other capacities. Students who learn to program can also disassemble and adapt components (Azevedo, in press). Having a single locus of competence also means that we do not have to decide in advance on any particular model of social factoring. We do not have to decide in advance, for example, whether the LaDDER model will work, but only when secondary developers do all the work of adapting or creating resources in schools. We can see how much teachers and students can contribute to this process without moving to different technology.

We emphasize that the cost-benefit trade-offs for teacher and student acquisition of technical competence are not transparent in the current educational computing scene. That almost everyone might drive and automobile, or that universal literacy might become the norm in modern civilization would have been impossible to foresee. Our project, more than most, aims to experiment with technology that does not foreclose at least the possibility of the broadest impart of technical competence.

CONCLUSION

From the Boxer Project's perspective, components are not a panacea or universal prescription for success in educational computing. In fact, the idea of components, by itself, may miss both of our fundamental commitments: (1) important constituencies such as teachers and students may be shut out of autonomous creativity. (2) Fluent transitions among and redefinition of levels (inside and outside components), such as afforded by a top-to-bottom uniform medium, may be foreclosed or forever isolated within particular communities; competence with technology may have an arbitrarily enforced ceiling. Instead, for us, components obviate the anachronistic assumption that tends to be made, explicitly or implicitly, among advocates of programming for widespread consumption: that everyone should make everything for themselves.

ACKNOWLEDGEMENTS

This work was supported by a grant from the National Science Foundation, number REC-9973156, to Andrea A. diSessa. The PI of the sponsored work reported here has a financial interest in PyxiSystems LLC, which is the owner of the Boxer software.

REFERENCES

References are included in "Issues in Component Computing."