Teaching the Design and Development of Educational Technology

Jochen Rick
Department of Educational Technology
Saarland University
D-66123 Saarbrücken
j.rick@mx.uni-saarland.de

ABSTRACT

The Department of Educational Technology at Saarland University is preparing an interdisciplinary master program for students with either a background in social or computer sciences. In the third semester, I will teach the required class "Programming for Educational Technology." In that class, students will learn how to design and develop robust, effective, and usable educational technology. The subjects covered in the class include software engineering, usability, and design, with an emphasis on interaction design for children. In addition to this ambitious learning agenda, the educational technology that students create should be able to serve as the basis for their master's thesis in the fourth and final semester. This paper describes the challenges involved in this course and some preliminary ideas of how to address those challenges.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*user-centered design*

General Terms

Design, Human Factors

Keywords

educational technology, interdisciplinary collaboration, computer science education

1. EDUCATIONAL TECHNOLOGY

The central tenet of modern educational theory is that learning is most effective when learners can actively explore subjects [2]. New technology, for its ability to fundamentally change human processes, has often been greeted as the vehicle to realize this pedagogy on a large scale [12]. Unfortunately, these visions have often fallen far beyond their grasp. In the 80s, Papert [11] cast the personal computer as the device to allow learners to take charge of their own learning. While that vision was compelling, it did not come to fruition for a variety of reasons [10]. In the 90s, the USA put a major focus on networking the classroom and connecting it to the wealth of information and resources that is the Internet. Again, this vision fell far short of revolutionizing learning [1]. Recently, there has been a major effort to add electronic whiteboards to classrooms in the UK; however, a progress report warns that the technology tends to "reinforce a transmission style of whole class teaching in which the contents of the board multiply and go faster, whilst pupils are increasingly reduced to a largely spectator role" [7].

Each of these failures shares a common pattern. First, new technology (respectively, personal computers, the Internet, and electronic whiteboards) arrives. Second, the public becomes enamored with it as the technology of the future. Third, there is a strong early push to introduce this technology into the classroom. Fourth, the technology enters the classroom without an established curriculum or supportive software. Fifth, because of this flawed approach, initial results are disappointing. Sixth, the public loses interest and the potential of the technology is never realized. The great challenge of educational technology is to disrupt this pattern.

2. AN INTERDISCIPLINARY MASTER

For technology to benefit education, it needs to be accompanied by appropriate software and curricula. Creating such support requires programming acumen, knowledge of modern pedagogy, ability to work with learners, and design skills. There are few people with the skill set to accomplish this by themselves. More realistically, interdisciplinary teams are needed. While there are enough people educated in human-computer interaction and enough people educated in education, interdisciplinary work can be challenging. Each field has different values and vocabulary. Computer scientists need to understand enough education to make appropriate design choices. Social scientists need to understand enough programming to set realistic goals. Both need to learn to collaborate across disciplines as part of an iterative design process. The new master program in educational technology at Saarland University is designed to promote such interdisciplinary collaboration.

The program will accept students with a bachelor's degree in either computer science or social science (psychology or education). While the majority of classes will be shared, a few will be geared to the background of the students. Computer scientists will take two introductory classes on empirical methods. Social scientists will take two introductory classes on programming. In the third semesters, the groups will be combined again for two classes that respectively cover empirical methods and programming in respect to educational technology. Students will learn to work together to design, develop, and evaluate innovative educational technology. In the fourth and final semester, they will put that knowledge to use for their master's thesis work. Ideally, the project work that students create in the third semester should serve as a basis for that

Position Paper for "Teaching Interaction-Design & Children" Workshop, IDC 2011

thesis work.

3. PROGRAMMING FOR ED. TECH.

In the first two semesters, students will gain an understanding of the theory and practice of modern educational technology. They will also gain the basic programming skills to contribute to the development and the basic statistics skills to contribute to the evaluation of educational technology. In the third semester, they will use those skills to collaboratively design and develop robust, effective, and usable educational technology in the "Programming for Educational Technology" class. While students will have a solid foundation, the class will cover additional topics in the fields of software engineering, usability, and design.

3.1 Software Engineering

Since students are expected to create a useful piece of educational technology, it must work. In other words, it needs to be robust enough not to crash and implemented well enough to function according to specification. Since these will be small-scale applications, the software engineering demands are not that high; only a few simple practices, such as working out an object model beforehand and creating a testing plan, should ensure the quality of the product. In addition, as a learning goal, students are expected to work in groups, with all team members contributing some to the code. To enable this, students will learn to use pair programming.

3.2 Usability

Design is a process of *reflection in action*—designers iterate between phases of implementation and evaluation [19]. Student designers must be able to evaluate their program for both its usability and usefulness (*i.e.*, learning potential in this case). While the first two semesters should have prepared them to evaluate the latter, the course will provide additional guidance for the former. Students will learn how to apply general principles of usability [9] to critique a design. As the vast majority will be working with children as the intended audience, they will need to learn the specific usability needs of children [4, 6] at different stages of development.

3.3 Design

The central principle of user-centered design is that users should be brought in early and often into the design process, so that significant efforts are not wasted on implementing bad design [8]. An established technique that students will learn is using low-fidelity prototyping [18] to test out ideas before they program them. By working closely with end users, designers can get a better understanding of their specific needs; however, this task becomes more difficult when the end users are children. Designing for (and with) children is particularly challenging as they tend to view the world differently than adults [3]. Effective adult evaluation techniques, such as think aloud protocols [8], may not be viable. For developers not intimately familiar with children of that age, it is often difficult to judge whether an activity is too advanced or alternatively patronizing [17].

While learners are users, they are a specific type of user with specific needs (*e.g.*, learners are novices in the application domain, not just the application interface) [20]. There are also design methods, such as *medium-based design* [16], that are specific to designing educational technology. Students will receive an introduction to these peculiarities of designing educational technology.

3.4 Challenges & Tensions

For a semester-long class that meets for only two hours a week, that's a pretty ambitious learning agenda. Obviously, not everything can be covered in depth. What gets priority? How much time should be allocated to studying the relevant literature? How much time should be allocated to supporting hands-on project work?

While we are set on the idea of group work, the details are tricky. Should groups be interdisciplinary? If so, the computer scientist could take the lead on programming and less time would need to be devoted to software engineering. On the other hand, an uneven pairing might lead to the task being split up, with the programming allocated to the computer scientist and the design and evaluation work allocated to the social scientist. This would run against the intent of the master program to expose social scientist to computing practices (*i.e.*, programming) and *vice versa*. Thus, there is a tension between creating a better product versus creating a better learning experience. Since the product is intended to be used for their master's thesis work, it has more value than in traditional design classes. Unfortunately, we still do not know who will apply for this degree and how the ratio of computer scientist to social scientists will play out. All interdisciplinary groups may not even be possible.

Another tension is between allowing flexibility versus supporting a specific approach. Educational technology is a large and diverse field. Good thesis work could be done on a large variety of projects. On the one hand, we would like to offer the students the possibility to choose a project that they are interested in. On the other hand, it could be very difficult to provide relevant support for a variety of projects. Forcing students to choose similar projects (utilizing the same technologies, targeting similar end users, using a similar design process, *etc.*) would allow for students to receive relevant support *en masse*.

3.5 Approach

Here is my current approach to the course: As I will also teach the second preparatory computing class to social scientist, I will start them off in Squeak [5], a cross-platform open-source Smalltalk implementation, which has been successfully used for a variety of high profile educational technologies, including Scratch [13]. Squeak has a particularly flexible graphical user interface, which can be simplified to better allow novices to engage such concept. Then, when the social scientists enter the "Programming for Educational Technology" class, they will be given support for continuing their work in Squeak.

One of the foundations of the German academic system, going back to Alexander von Humboldt, is to unite teaching and research. So, students are encouraged to contribute to the research agenda of the department. One of the two major research strands of the department is supporting co-located collaborative learning with interactive surfaces. Given Squeak's inherent flexibility, it is a good environment for adapting to these new technologies. For instance, I have used Squeak to develop a variety of applications for the DiamondTouch interactive tabletop [14] and the Apple iPad tablet. While both have different native development environments, I was able to create middleware that allowed Squeak to run on both devices. I plan on providing such middleware along with a framework for developing multi-touch applications to the students. Thus, they will be well supported in creating applications for these novel platforms.

This approach has several benefits. First, as social scientists will already have exposure to Squeak in the preparatory class, they will have some programming competence to contribute to their team, even if they are partnered with a computer scientist. Second, low fidelity prototyping is a particularly compelling method for developing applications for interactive surfaces [15]. Third, the co-located collaborative setting makes it more likely that students will verbally share their understanding with their partner. This dialogue will serve as a good basis for both formative and summative evaluation.

To further support students, I will provide one or two high-quality sample applications that students can use as concrete models of how to design such applications. Pairs of social scientist might even just slightly modify these sample applications to create their educational technology (placing more emphasis on the surrounding curricular design and evaluation).

4. THIS WORKSHOP

While this is my current approach, it is over a year until this class will be taught for the first time. While preparatory work will be necessary to support this approach, it is still early enough that many aspects of the class can be reexamined. While I have some experience in designing technologies for children, I have yet to teach it to others. I hope this workshop will provide me with additional insight about how to incorporate vital components of interaction design for children into this class.

5. REFERENCES

[1] A. Bruckman. The day after Net Day: Approaches to educational use of the Internet. *Convergence*, 5(1):24–46, 1999.

[2] A. Collins, J. S. Brown, and S. E. Newman. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick, editor, *Knowing, Learning, and Instruction: Essays in honor of Robert Glaser*, pages 453–494. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[3] A. Druin. Cooperative inquiry: Developing new technologies for children with children. In *Proceedings of CHI '99*, pages 592–599, New York, 1999. ACM Press.

[4] A. Druin. The role of children in the design of new technology. *Behaviour and Information Technology*, 21(1):1–26, 2002.

[5] M. Guzdial and K. Rose, editors. *Squeak: Open Personal Computing and Multimedia*. Prentice Hall, Upper Saddle River, NJ, 2002.

[6] P. Markopoulos, J. C. Read, and S. MacFarlane. *Evaluating Children's Interactive Products: Principles and Practices for Interaction Designers*.

[7] G. Moss, C. Jewitt, R. Levačić, V. Armstrong, A. Cardini, and F. Castle. The interactive whiteboards, pedagogy and pupil performance evaluation: An evaluation of the schools whiteboard expansion (SWE) project: London challenge. Research Report 816, Department for Education and Skills, Institute of Education, London, 2007.

[8] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993.

[9] D. A. Norman. *The Psychology of Everyday Things*. Basic, New York, 1988.

[10] R. Noss and C. Hoyles. *Cultures and change*. Kluwer, Boston, 1996.

[11] S. Papert. *Mindstorms: Children, Computers and Powerful Ideas*. Basic, New York, second edition, 1993.

[12] N. Postman and C. Weingartner. *Teaching as a Subversive Activity*. Dell, New York, 1969.

[13] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[14] J. Rick. Six applications for interactive tabletops. In *"Collaborative Learning with Interactive Surfaces: An Interdisciplinary Agenda" workshop, ICLS 2010*, 2010.

[15] J. Rick, P. Francois, B. Fields, R. Fleck, N. Yuill, and A. Carr. Lo-fi prototyping to design interactive-tabletop applications for children. In *Proceedings of IDC '10*, pages 138–146, New York, 2010. ACM Press.

[16] J. Rick and K. K. Lamberty. Medium-based design: Extending a medium to create an exploratory learning environment. *Interactive Learning Environments*, 13(3):179–212, 2005.

[17] J. A. Rode, M. Stringer, E. F. Toye, A. R. Simpson, and A. F. Blackwell. Curriculum-focused design. In *Proceedings of IDC '03*, pages 119–126, New York, 2003. ACM Press.

[18] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *Interactions*, 3(1):76–85, 1996.

[19] D. A. Schön. *Educating the Reflective Practitioner*. Jossey-Bass, San Francisco, 1987.

[20] E. Soloway, M. Guzdial, and K. E. Hay. Learner-centered design: The challenge for HCI in the 21st century. *Interactions*, 1(2):36–48, April 1994.