Propagating Software-Based Educational Innovations

Edward F. Gehringer[1]

Abstract – Much software has been created for teaching various aspects of computing. Most of it, however, is sparsely used. Educators frequently do without any of it, or "reinvent the wheel." This paper discusses approaches to improving the visibility and adoptability of educational software. First, others must be aware of the software. Conferences can play a major role. Work can also be presented in a poster or a demo, or by organizing a panel or a BoF on a related topic. Publishers can also help, by publicizing your work on Web sites associated with their textbooks. Once aware of the innovation, faculty must be reminded, e.g., by e-mail, at the appropriate time in their course. Try to minimize the need for risk-taking, especially for software without a long track record: If it can be used in a short assignment, instructors will be more inclined to adopt and expand usage later.

Keywords: Educational applications, Dissemination.

INTRODUCTION

As educators teaching about technology, we are well aware that technology can improve our teaching. Many of us are involved in developing such technology. The National Science Foundation makes about 100 awards a year in its Course, Curriculum, and Laboratory Improvement (CCLI) program. Virtually all of the computer-science awards involve some sort of software development, as well as many of the awards in other fields.

Yet just as publication of an article does not insure it will be read, much less cited, development of software does not insure that it will be used, much less disseminated to other institutions. For example, in 2006, the author had a student go through the proceedings from the Eclipse Technology Exchanges [1] in an effort to find Eclipse plugins that could be used in an o-o design course. Of the 50-odd tools described, only two of them turned out to be useful. Many required a specialized environment in which to run. Some required extensive hand-tuning to come up with reasonable results. Some no longer worked in the current version of Eclipse. It was quite surprising that in a situation where we sought out tools to reuse, we could find so few.

Even assuming a good match between the software developer's aims and the instructor's needs, barriers remain. First, the instructor must know about the software. Second, the software must fit into the instructor's class without requiring "too much" change in style or substance. Third, the documentation needs to be sufficient for the instructor to figure out how to use it with ordinary effort. Finally, the developer needs to provide enough support to make the instructor comfortable with the software, and to fix any problems that may arise. In subsequent sections, we will consider each of these hurdles.

THE HISTORY

The author is the developer of a project called Expertiza. Initially it was a peer-review application oriented toward producing reusable learning objects by a divide-and-conquer approach [2] (each student or team chooses a small part of a large project, and competes with other students/teams to produce the best version of that part). It has grown into a more general platform for collaborative learning, supporting file, Web-page, and wiki submissions and several

[1] Department of Computer Science and Dept. of Electrical & Computer Engineering, North Carolina State University, Raleigh, NC 27695-8206, efg@ncsu.edu

different kinds of reviews. The current version is a Web-based application written in Ruby on Rails. It can run on any platform, and has been installed at two sites in Europe as well as our site in the US.

Starting in 2005, outside instructors have been invited to use the system in their classes. They have been made aware of the system primarily by talks given by the author (16 talks between 2006 and 2008 at professional conferences, 2 panel presentations and 5 poster presentations, as well as twelve talks on various college campuses). During this period, over 300 faculty have signed up to be informed of progress on the project, and approximately three dozen have indicated their intention to use the system in their classes. To date, however, only 19 other instructors have used it, and most of those have used it for only one semester. It is instructive to investigate why so much interest has led to so little use. Toward that end, 25 "intenders" among the instructors were surveyed in June 2008. Eleven responded, for a rate of 44%. In addition, four instructors who did not fill out the survey sent e-mail comments; including these yields an overall response rate of 60%. Their answers give some insight into why this project—or any project offering innovative approaches to teaching—tends to meet resistance.

BARRIERS TO ADOPTION

Instructor Reticence and Inertia

The first hurdle is instructor reticence. It takes effort to introduce an innovation into one's teaching. An instructor will be willing to undertake that effort only if (s)he expects to reap some benefit from the innovation. As one of our survey respondents put it,

> "No matter how much good documentation, there is an attitudinal issue about why any instructor would take on the additional workload of incorporating a new system into their already daunting task of teaching the number of students they have … A similar thing happens when there is an implied message that the administration would like to see "online courses" -- most faculty think that hybrid/online is here to stay but many of them see it as an unfunded mandate for them to change their mode of instruction."

The experiment might even backfire if the students are confused, or resistant to the innovation. While the Expertiza project has published papers on positive results [3, 4], these might be described as anecdotal. Thus, adopters tend to be concentrated among those instructors who already believe in collaborative learning. Most college and university faculty are quite cognizant of research in their academic field, but rather ignorant of research into teaching methodologies. Thus, the collaborative-learning believers tend to be a small group.

Student Resistance

Any software application has a "learning curve." Even if the application makes it easier to learn the course material, from the student's point of view, there are now two things to be learned: The application, and the subject that the application covers. The issue may become whether learning the application *plus* the material is harder than learning just the material, without the application. This is less of an issue for computer-science students than it is for others. In 2005, we found, via a focus group, that a user interface that posed no problems for the author's computer-science students was seen as a major hurdle by students in a zoology class. These problems might diminish as our classes become increasingly populated by digital natives; however, students' expectations of UI functionality are increasing.

Student resistance can lower morale and thus get in the way of learning. It can also hurt in another way. As one of our respondents points out,

> "'New' approaches are new for the student as well as the instructor, and unless the instructor really pushes the system really hard, there's going to be student resistance which might show up in final reviews."

Ours is not the only electronic peer-review system to encounter student renitance. Keeney-Kennicut et al. [5] report the same phenomenon using Calibrated Peer review [6], with a majority of students in the first semester saying that the system should be discontinued. But as they gained experience with the system, student perceptions improved.

Institutional Considerations

Instructors frequently feel constrained by the perceived wishes of their institution. This may include adopting a particular learning-management system (LMS), such as Blackboard or Moodle. If so, they may favor resources that are integrated with the LMS and disdain those that do not fit into the mold. As one instructor put it,

> "The number 1 reason why I didn't use the system is that I teach older adult students and it seemed to be a burden for them to learn Vista/Blackboard along with a separate technology."

At smaller colleges and high schools, anything that requires more technology may be a stumbling block. One of our instructors in 2007 was at a small college whose network capacity was inadequate to handle demand. Only the students who had off-campus ISPs were able to use the system successfully. Since most of his students were residential, he had to drop out of the program.

Suitability for Course and Teaching Style

Any software application presupposes a particular way of covering the material. Some instructors simply cover the material in a different way. Our Expertiza application uses peer review. Instructors whose homework consists of problem sets cannot effectively use it. Another instructor told us that he runs design competitions for his final project, and students do not want to submit their ideas for peer review, since other students would then be able to "steal" them.

One might, of course argue, as the accrediting agencies have, that assigning only problem sets is not an effective way to teach students what they need to know to be effective on the job. Regardless, the fact remains that type of work assigned is an additional reason for instructors not to adopt a tool.

Usability and Documentation

As long as the developer uses a system solely in his/her own classes, only student documentation is needed. But in many systems, the instructor needs to perform many tasks the students don't. Early development tends to concentrate on the student UI and student documentation. The instructor knows the system intimately, having designed it, so little effort is expended on the instructor UI and documentation. This makes it very difficult for other instructors to get acclimated. Moreover, the instructor UI and documentation are much larger tasks than the student interface. Until recently, we encouraged instructors to let *us* enter the assignments. But this diminished the other instructors' sense of ownership and ability to make updates quickly. One told us that he dropped the system because he had to ask us to perform trivial operations, such as adding a new user.

Support

Software support is a concern beginning on Day 1, and lasting as long as the system is in use. In the early going, most questions relate to bugs, which must be fixed quickly. As the system grows more complex, questions concern functionality, or misperceptions of functionality.

As a system grows larger, it is difficult to pay for support on a research grant, and it is probably not something for which college credit can routinely be given. Therefore, documentation and help facilities become much more important. Videos are rapidly growing in popularity as a documentation medium. They can be recorded in commercial applications such as Camtasia, or with open-source screen recorders such as Camstudio (http://camstudio.org).

WHAT'S MOST IMPORTANT?

In our survey, we posed ten reasons to our adopters on why they did not try, or did not continue with, Expertiza:

- I had trouble learning to use the system.
- I would have felt more comfortable if I had seen more documentation on what the system could do.
- I tried to use the system, but had trouble with the user interface.
- I would have felt more comfortable if I had seen more documentation on how to use the system.
- I was unable to get enough help from the Expertiza staff.

- I was concerned that my class might have trouble using the system.
- I had trouble with bugs in the software.
- I was requiring my students to use other software applications, and did not want to introduce too much technology in one semester.
- I had trouble thinking of a suitable assignment to use with the system.
- I intended to use the system, but just ran out of time.

Responses were obtained using a 5-value Likert scale (strongly agree, agree, neutral, disagree, strongly disagree). Respondents had an opportunity to add a prose comment on any question.

Easily the most important reason (see Table 1) was the need for more documentation, with all but one instructor agreeing, and half the instructors strongly agreeing. Next was doubts about student reaction, especially among the non-CS/IT instructors. The only other important reason was "just running out of time."

At the other end of the scale, the instructors said they had no difficulty getting help from our staff, or thinking up assignments for which peer review could be used to build resources useful to others. The latter was a surprise, since it is often not easy to think of reusable learning objects that a class is capable of creating; this was discussed in a 2006 paper [7]. It may indicate that these instructors are thinking only of how to use peer review, not using it to create learning objects to enhance other classes' learning.

Table 1. Results from Instructor Survey

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Mean (5=SA, 1=SD)
I had trouble learning to use the system.	1	1	5	2	1	2.9
I would have felt more comfortable if I had seen more documentation on what the system could do.	1	5	0	2	2	3.1
I tried to use the system, but had trouble with the user interface.	1	3	3	0	2	3.1
I would have felt more comfortable if I had seen more documentation on how to use the system.	5	4	1	0	0	4.4
I was unable to get enough help from the Expertiza staff.	0	0	2	4	3	1.9
I was concerned that my class might have trouble using the system.	4	2	1	0	1	4.0
I had trouble with bugs in the software.	0	1	6	0	0	3.1

VENUES FOR DISSEMINATION

How does one make other instructors aware of an educational software innovation? There are several choices.

First, the innovation can be announced on an **e-mail list**, such as SIGCSE-members@acm.org, which has over 1000 members. This is useful if your target group includes computer-science educators. This tells readers less than they would find out by listening to a talk—but you could always post a link to a video that explains what a talk would.

Second, you could contribute the innovation to **CITIDEL** (the peer-reviewed Computing and Information Technology Interactive Digital Educational Library). Interested instructors can sign up to receive e-mail updates, but likely SIGCSE-members will reach a far larger audience at the present time.

Presentations at **conferences** are valuable, such as SIGITE, SIGCSE, ITiCSE, ICER (International Computing Education Research workshop), the ASEE conferences, the CCSC (Consortium for Computing Sciences in

Colleges) conferences, or the ACM regional conferences. Most of these provide several ways to be involved, e.g., by having a paper accepted, or a poster, or a panel proposal. You might, for example, organize a panel on various approaches to teaching a subject that your software is designed for. Or you might organize a "birds-of-a-feather" session (BoF) on the topic; these are less competitive than panels, but also tend to attract a smaller audience.

In addition, there are a number of conferences devoted to technology in teaching; these are especially useful if your innovation is not discipline specific. They include the MERLOT International Conference, the SALT (Society for Applied Learning Technology) conferences, the Sloan-C conferences (these are really about distance education, but most educational technology is useful in distance ed), and the National Educational Computing Conference, which is a huge convention attended mostly by K-12 educators. At these conferences, one can do a presentation without writing a paper; thus they may help you promote your innovation, but they will not help you pad your publication record for promotion.

Many **journals** have a wide enough readership that they can serve as a dissemination vehicle. Among these are the *Journal of Computing Sciences in Colleges*, which publishes the proceedings of the CCSC conferences, the *ACM Journal of Educational Resources in Computing*, the *Journal of Engineering Education*, and the *International Journal of Engineering Education*. A widely read journal on educational technology is the online journal *Innovate* (http://innovateonline.info). *Innovate* authors are given the opportunity to do a Webcast on their articles. The Webcast is archived and can be used to introduce others to the author's work.

Textbook publishers should not be overlooked. They may be willing to make your project an ancillary resource for their textbook, or at least to publicize it on the textbook Web site. Probably the best way to contact publishers is to meet the editors on the exhibit floor at conferences such as SIGCSE or the ASEE Annual Conference and Exposition.

FOUR SUCCESS STORIES

This final section of the paper relates the experiences of four research projects on educational software that are making, or have made, the transition to being widely used in classes at many institutions.

WebAssign started about 1997 as a research project in the Physics department at North Carolina State University. It was a system for doing testing over the Web. The authors obtained a large Department of Education grant to expand the program to disciplines other than physics. Early on, the authors began working with textbook publishers to input questions from textbooks, so that the textbooks could be advertised as "Webassign-enhanced," and homework assigned from them could be graded electronically (the system allowed for randomized parameter values, so each student would get a different answer). Eventually the project needed to hire a sales force to continue their growth. This was very difficult to do within the (nonprofit) university. So in 2003, the project became a company, after purchasing their computers and other assets from the university. In Spring 2008, 226,000 students from 1241 schools used the system.

DyKnow began about 2003 as a research project by Dave Berque in the Department of Computer Science at DePauw University. It is a client-server system that allows students and instructors using tablet computers to communicate with each other. The instructor can pose a question to the students, who can answer it on their tablet computers. Then the instructor can view or display any student answer anonymously for all to see.

The prototype application generated "some good press," which caught the eye of a DePauw alumnus with a long track record of starting high-tech businesses. He assembled a team to commercialize DyKnow. In Spring 2008, the application was used at about 200 schools in six countries.

Web-CAT is an automated grading application for programming assignments, originated by Steve Edwards in the Department of Computer Science at Virginia Tech. It was funded as a CCLI project. The authors gave talks at conferences such as SIGCSE and CCSC. They found enthusiastic users on 4 to 6 other campuses, which helped them win CCLI Phase II funding. Web-CAT is now used in about 30 schools.

JHavéPop, developed by David Furcy in the Department of Computer Science at the University of Wisconsin-Oshkosh, is a much smaller and newer project than the other three. It is a mechanism for teaching programming of linked lists in Java. The instructor displays a diagram of a linked list and asks the students to write statements to perform some manipulation of the list (e.g., to add, remove, or rearrange particular nodes). It was first announced by e-mail to the SIGCSE-members list on January 23, 2008. It was also promoted at a SIGCSE BoF in March. It has been used at use at five other colleges and one high school. A total of 666 distinct users have visited the site (this includes students as well as instructors).

All of these projects took different routes to visibility and dissemination. The author hopes that the suggestions in this paper will help others to achieve success for their projects.

SUMMARY

In order to disseminate an educational software project, others must be made aware of it, and it must be accessible to them. Projects can be publicized via e-mail lists, digital libraries, conference presentations of all kinds, and journals. Textbook publishers have often played a crucial role in making their adopters aware of software innovations. One of the most important factors in disseminating a project is insuring that it is easy to use. Great care must be taken in designing a UI. Documentation, especially instructor documentation, needs to be written. Often students may fail to see the worth of a new innovation, and react unfavorably to having to use it. This suggests that instructors need to concentrate on showing students how it will help them.

REFERENCES

[1] M. Robillard, "Eclipse Technology Exchange Workshop at OOPSLA 2007," http://www.cs.mcgill.ca/~martin/etx2007/

[2] E. F. Gehringer, L. M. Ehresman, S. G. Conger, and P. A. Wagle, "Reusable learning objects through peer review: The Expertiza approach," *Innovate—Journal of Online Education* 3:6 (August/September 2007).

[3] E. F. Gehringer, L. M. Ehresman, and D. J. Skrien, "Expertiza: Students Helping to Write an OOD Text", OOPSLA 2006 Educators' Symposium, Portland, OR, October 23, 2006.

[4] E. F. Gehringer, "Assessing students' wiki contributions," 2008 ASEE Annual Conference and Exposition, American Society for Engineering Education, Pittsburgh, PA, June 22–25, 2008.

[5] W. Keeney-Kennicutt, A. B. Gunersel, and N. Simpson, "Overcoming student resistance to a teaching innovation," *International J. for the Scholarship of Teaching and Learning* 2:1 (Jan. 2008).

[6] R. Robinson, "Calibrated peer review," *The American Biology Teacher* 2001, pp. 474–480.

[7] E. F. Gehringer, L. M. Ehresman, S. G. Conger, and P. A. Wagle, "Reusable learning objects through peer review: The Expertiza approach," Frontiers in Education 2006, San Diego, October 22-26, 2006.

Edward F. Gehringer

Edward Gehringer is an associate professor in the Department of Computer Science and the Department of Electrical and Computer Engineering at North Carolina State University. He has been a frequent presenter at education-based workshops in the areas of computer architecture and object-oriented systems. His research interests include architectural support for memory management, garbage collection, and computer-supported collaborative work. He received a B.S. from the University of Detroit(-Mercy) in 1972, a B.A. from Wayne State University, also in 1972, and the Ph.D. from Purdue University in 1979.