AN ELECTRONIC INFRASTRUCTURE FOR K-12 SCHOOLS

Written by Steve Casburn
LIS 385T.6 – Andrew B. Whinston

Submitted: 2 December 1998

Introduction

The authors of the paper “An Electronic Infrastructure for a Virtual University” [1] ably address the question of how people with several thousand dollars to spend could build an electronic infrastructure that would “exploit the full potential of [networking] technology” in order to “[reengineer] the educational process itself.” This paper will briefly examine a somewhat different question: How such an electronic infrastructure might be built for public K-12 schools, which generally have little discretion over how their budget is allocated.

The infrastructure proposed here would be for a physical school rather than a virtual university. It would carry standard (though customizable) K-12 curricular materials which would be free, rather than cutting-edge materials which would be costly. It would run using free software on old computers rather than proprietary software on new computers. In a word, this infrastructure would be inexpensive.

Though Americans as a whole spend a great deal of money (about $320 billion per year, or 4% of the GDP) on K-12 education, the current market for educational software is relatively small (about $640 million per year) [3] and few companies are making any money producing it. Schools, which should be the big market for educational software, are reluctant to spend money on it because commercial software is rarely designed to fit into the school's curriculum and can rarely be customized to fit into it. Faced with these limitations, schools spend their scarce dollars on curricular materials that return more value. [9]

From the standpoint of public K-12 schools, then, the ideal educational software would be free and customizable. Surprisingly, a secure foundation for developing such software exists: the open source software movement. Open source software programs are free, non-proprietary programs whose uncompiled source code can be downloaded and modified by the user. The expanding of the Internet in 1992-94 has led to a boom in open source software development, as distribution of and feedback about open source programs has become considerably faster and easier. [7]

This paper will provide a description of the open source development paradigm, then address two separate questions. First, how might educational software be developed using the open source method? Second, can schools save money on computer hardware by taking advantage of existing open source operating systems and applications?

The Open Source Paradigm

The philosophy behind open source software can be stated simply: Hundreds of software programmers coding independently for fun can be (and often are) much more collectively productive and brilliant than dozens of software programmers coding together under contract for a salary. Open source project leaders (or, “moderators”) facilitate development by coordinating programmers’ contributions and waiving legal proprietary rights in the project. The motive for this surrender of rights was described by Open Source Initiative president Eric Raymond: “We don’t want power and we don’t care much about money *per se*. What makes us run is solving problems and inventing things.” [8]

Open source projects typically begin when a moderator advertises a program he has written or inherited, explains its purpose, and solicits the help of “beta testers” -- people who are willing to seek out and help fix any bugs in the program in exchange for having a better program and being recognized for their programming skills. When a beta tester finds a bug in the program, has a possible bug fix, or wants to request a new feature, he informs the moderator, who then decides whether to accept the suggestions. If the changes to the program are significant, the moderator updates the program’s version number, then sends the revised program out to the beta testers for another iteration of testing and suggesting. Open source programs are always free, and most significant programs can be freely downloaded through the Internet from any of several software archives. [5,6,7]

Because of the informal nature of open source projects, compiling a track record of the success or failure of the genre as a whole is difficult, and perhaps impossible. It can be said, though, that the open source concept remains immensely respected among computer programmers, and that three of the most notable software successes of the 1990s -- the Linux operating system, the

Apache Web server, and the Perl scripting language -- are open source projects. [6,7]

Developing Open Source Educational Software

As successful as it might be off-campus, the open source paradigm must be modified before being introduced on-campus. The major problem that the paradigm faces in schools is that the people with pedagogical experience (the teachers) usually can't write code, and the people with code-writing skills (computer-savvy students) usually lack pedagogical experience.

The key to making open source educational software development work, then, is forging a partnership between teachers and computer-savvy students. Such a partnership would be a win-win situation: good for the teachers, because they get customized software; and good for the students, because they will get real-world programming experience and finally have something interesting to do in school.

Here is how such a software development process might work: A teacher gets an idea about a kind of program he would like to use, and searches existing educational software archives on the Internet to see whether such a program already exists. If one does, he downloads the code and works with a student to customize it. If one doesn’t, he describes what he needs with a student, and the student writes up a simple program that will do what the teacher needs. (Obviously, the teacher will have to think well ahead of time about what sort of programs he will need when.)

In keeping with the open source tradition, new programs would be posted to the major educational software archives and significant modifications to existing programs would be mentioned to the moderator of that program. If a new program elicits enough interest, then the student who wrote it could choose to become its moderator and lead it on to further development and wider use.

Using Existing Open Source Programs in Schools

People who develop open source software tend to like to set up and work with open source software, a fact that could save schools who move to an open source model tens of thousands of dollars in computer hardware costs. A commercial operating system such as Windows 98 usually requires relatively new hardware to run at a workable speed. The Linux operating system, on the other hand, runs smoothly on computers with Intel 486 or PowerPC 601 processors.

Because Linux runs well on machines that are several years old, a school that sets up its computer labs to run open source software can get its hardware for free, or almost free. Corporations that are upgrading their computer hardware to run the latest version of Microsoft Windows and Office have plenty of now-useless (to them) Intel 486 computers that they would be willing to give to schools in exchange for a tax deduction. A school could set up an open source computer lab for little more than the cost of running the necessary wiring.

Students using such a lab would be able to surf the Web (Netscape has been ported to Linux), set up a Web server (Apache is the most common Web server application used for heavy-traffic Web sites), write Web scripts (Perl is a popular scripting language), and run DOS, NeXTstep, and Java programs in emulation. From the standpoint of a student user, the result is hardly less functional (and far more reliable) than a Windows 98 machine; from the standpoint of the school, the result is an example of that magical word, “free.”

Conclusion

Obviously and famously, there’s no such thing as a free lunch. Schools which adopt the open source paradigm will trade off time, space, ingenuity, and adult supervision in order to save money. However, in the short term, these resources are more likely to be elastic and re-allocable than money would be. Open source software and hardware would cost schools almost no money in the short term, and are the obvious means by which they can mount an ambitious effort to use the Internet to “reengineer the education process itself.”

References

1. Chellappa, Ramnath, Anitesh Barua, and Andrew B. Whinston. “An Electronic Infrastructure for a Virtual University.” *Communications of the ACM 40*, 9 (Sept. 1997), 56-58.

2. Hämäläinen, Matti, Andrew B. Whinston, and Svetlana Vishik. “Electronic Markets for Learning: Education Brokerages on the Internet.” *Communications of the ACM 39*, 6 (June 1996), 51-58.

3. Martin, Justin. “Lifelong Learning Spells Earnings.” *Fortune* (July 6, 1998).

4. Milbank, Dana. “Schoolyard Tussle.” *The New Republic* (December 14, 1998), 22-25.

5. Perens, Bruce. “The Open Source Definition,” version 1.0 (June 1997). Current version at: “http://www.opensource.org/osd.html”

6. Raymond, Eric S. “The Cathedral and the Bazaar,” version 1.40 (Aug. 11, 1998). Current version at: “http://www.tuxedo.org/~esr/writings/cathedral-bazaar/”

7. Raymond, Eric S. “Homesteading the Noosphere,” version 1.10 (July 11, 1998). Current version at: “http://www.tuxedo.org/~esr/writings/homesteading/”

8. Raymond, Eric S. “An Open Letter to AOL.,” (Nov. 29, 1998). Available at: “http://www.opensource.org/aol-letter.html”

9. Soloway, Elliot. “No One is Making Money in Educational Software.” *Communications of the ACM 41*, 2 (Feb. 1998), 11-15.