Transforming High School Classrooms with Free/Open Source Software: *It's Time for an Open Source Software Revolution*

Jay Pfaffman
University of Tennessee, Knoxville

Free/Open Source Software (FOSS) applications meet many of the software needs of high school science classrooms. In spite of the availability and quality of FOSS tools, they remain unknown to many teachers and utilized by fewer still. In a world where most software has restrictions on copying and use, FOSS is an anomaly, free to use and to redistribute without restrictions. Using FOSS instead of the ubiquitous proprietary applications most commonly used can seem as heretical and as impossible as did Copernicus's suggestion that the Earth was not the center of the universe. This paper suggests one of the factors that keeps schools from adopting—or even considering—FOSS is that its very existence can be written off as an anomaly. Additionally, it shows similarities between the development of FOSS and the development of scientific knowledge. Moreover, it provides examples of FOSS for high school science classrooms.

Paradigm Shifts

In 1980, a paradigm shift (Kuhn, 1962) arguably as dramatic as those initiated by Copernicus or Einstein, began in the world of software. Richard Stallman, a computer programmer, needed to modify the software to his new printer, as he had done with its predecessor. Previously, he had access to the program source code—the files used to create the program—which allowed him to improve and expand the software. This time he did not have the source code. When he finally found a colleague who did, his colleague could not share it due to a nondisclosure agreement, a new policy that surprised Stallman. Vexed by these unprecedented restrictions that kept him from making the best use of his equipment, Stallman snapped. Outraged, he felt that corporate greed had intruded upon a sacrosanct part of the computer-users' culture (Williams, 2002). Stallman responded by writing the GNU Manifesto,[1] proclaiming that software should be free of charge, and give everyone the unrestricted right to learn from it, use it, change and distribute it.

1 GNU is a self-referential acronym that stands for "GNU's Not Unix."

Stallman's manifesto remains largely unknown to high school educators whose limited budgets make the promise of free software especially germane. The work of Stallman and many who embrace his ideals has resulted in Free Open Source software (FOSS) which in many cases obviates the commonly used proprietary standards. In spite of this vast body of Free software, many people think that using software without paying is stealing. The idea that computer users have a fundamental right to redistribute the software they use stands in direct opposition to the commonly understood right of a software creator to restrict redistribution. Understanding a license that forbids *restrictions on redistribution* is difficult to understand when most software is distributed with a license forbidding redistribution.

Free/Open Source Software (FOSS) is based on the principle of freedom, not cost. Software that does not provide the source code is termed "proprietary." The inner workings of proprietary programs are secret, which makes it difficult for other programs to inter-operate with them, and impossible to fix problems with them. Proprietary software, even when available for no cost, restricts users' freedoms. Part of this notion of freedom comes from the *source code*—the human readable files used to create the program. The source code allows people not only to use the software, but also to learn from it or change it to better suit their needs. Just as freedom of speech benefits those who are not journalists or protesters speaking out against a government, source code benefits everyone, not just programmers. FOSS gives everyone the right to use and distribute software as they see fit.

This radical change is becoming recognized by an increasing number of instructional technologists, teachers, and high school administrators (Hepburn, 2005). FOSS promotes social justice practices in that it provides free software to end users and an efficient means to develop software for programmers. The ramifications of this practice of equity have significant implications for students and schools. The notion that software should be freely redistributable is important for schools because when a school chooses a particular software package it implicitly makes that decision for its students—students who want to work on computers both at home and at school.

Open Source Software Use and Development

The Internet is built on FOSS applications. The Berkeley Internet Name Daemon (BIND), the program that lets people use domain names rather than Internet Protocol (IP) numbers is FOSS. Apache, the most-used web server, is also FOSS. The most widely recognized Open Source development model is known as the Bazaar Model (Raymond, 1997). As opposed to the traditional "cathedral" development model, in which a small number of developers release "finished" products, often infrequently, the Bazaar Model has much more frequent releases to give developers feedback on their work more quickly. A key part of the Bazaar Model is to have two different versions of a program, a "stable" version that has fewer features, but is more reliable, and a "development" version that has new features that may not be completely bug-free. The development version benefits users who get new features and frequent bug-fixes, and developers, who benefit from feedback from users.

The viability of this Bazaar Model has been vetted by several companies embracing this model to attract a community of developers and testers for their software (Goldman, 2005). For example, Netscape released the source code to its Netscape Communicator package. Jim Barksdale, Netscape's president and CEO, explained the decision this way: "By giving away the source code for future versions, we can ignite the creative energies of the entire Net community and fuel unprecedented levels of innovation in the browser market." Netscape's decision resulted in Mozilla, a full-featured suite of software and, subsequently, the Firefox web browser. These Open Source programs continue to benefit Netscape's commercial products. Similarly, Google's servers run the FOSS Linux operating system; when Google's programmers find problems and their solutions, those solutions are given back to the community so that all may benefit from them.

FOSS as a Social Justice Concern

Though the GNU Manifesto is little known, and its marginalized, non-hegemonic status makes it inaccessible to most high school science educators, it has resulted in many useful tools for K-12 schools, and especially the science classroom. The fact that FOSS software is free gives it many practical advantages over the sometimes expensive programs that it can replace. Even for a school or classroom that has a large budget for software, having the freedom to install the software on all computers and send it home with students is a convenient time saver.

These advantages are not merely fiscal and pragmatic, they also have important implications for teachers and teacher educators who often become the unwitting sales agents of software companies. Software is made more valuable when more people use it, and hence when schools adopt it. For example, when teachers require students to turn in assignments using a proprietary file format like Microsoft Word's, this implicitly suggests that in order to be a successful student one must buy, know, use, a particular software program. Similarly, when teachers design courses that teach particular computer skills, basing these courses on particular proprietary products makes those applications an obligatory point of passage (Latour, 1987), that is, a requirement for being a full participant in the learning community. Teaching students to use a particular program, especially an expensive one for a specialized task adds significant value to that particular program. Consequently, the deep discounts that software companies give schools on virtually all software packages is not a function of their altruism, but good business sense. Students leave high school and college expecting to have the software tools they know available to them and if their employers do not provide them, they are virtually forced to buy them or, frequently, obtain illegal copies. Teaching students to use free tools empowers them to continue to use the skills they gain in school without being forced to choose between buying or stealing a program.

Teaching students about FOSS often and early and providing them with opportunities to access, work with, and produce assignments with FOSS will give students a broader range of decisions to make about how to choose computer software. For teachers and administrators, establishing a knowledge base about FOSS provides them with the ability to make better decisions. For example, by choosing a FOSS solution, a school or district can re-purpose those funds to hire staff to train teachers to more effectively integrate those tools into their instruction.

FOSS and Science

Science teachers should be particularly interested to understand that the development of FOSS is analogous to the development of scientific knowledge. Both FOSS and scientific knowledge are built on the concept of creating shared knowledge and the desire to have one's work adopted by the scientific or computer-using community. Both have historically been developed by amateurs and both are often advanced by having access to a network.

Science is predicated on open, free dialog and on sharing. Although an engineer might find a solution to a problem and keep it secret for a single company's gain, a scientist's goal, by contrast, is twofold: find a new discovery or explanation, and spread it to the entire scientific community. Therefore, scientists are willing to pay to publish their work in journals like *Science*. Similarly, FOSS developers want their work and ideas broadly disseminated (Kogut and Metiu, 2001). Because software is made more valuable when it is used by more people, giving software away has the potential to add value. For both scientists and FOSS programmers, sequestering one's work to only certain people is antithetical to their fundamental goals.

Another similarity between FOSS and science is that both are often avocations. Gregor Mendel, who established the field of genetics, was an abbot. Nicolaus Copernicus, who established that the sun revolves around the sun, never performed astronomical observations and calculations professionally. Albert Einstein worked as a patent clerk when he wrote the four papers that form the foundation of modern physics. Today, many scientists and FOSS

developers have the luxury of doing so as a vocation, but amateurs continue to contribute to the development of science in significant ways.

One group of amateurs widely recognized by the professional and academic communities as contributing significantly to the advancement of science is amateur astronomers. There are too many objects in the sky for professional astronomers to be able to see everything. Consequently, amateurs have been able to make significant discoveries — even with equipment vastly inferior to that of professionals (Ferris, 2002). Because not even paid astronomers have the luxury of gazing at the sky at every instant, for decades many universities and laboratories have shared their telescopes with amateurs. In a recent change, the Internet now allows professional astronomers to share data with anyone who will look at it. The Internet Space Station Amateur Telescope (ISSAT) project proposes to put a telescope on the International Space station to be controlled by amateurs (Beatty, 2003). Advances in computing digital imaging have afforded this community of amateurs tools that can bring into people's homes telescopes that previously would have cost tens or hundreds of thousands of dollars. As a result, amateurs are contributing new discoveries to the field of astronomy at an ever-growing rate.

Another connection between FOSS and science is the importance of a network. Meteorology became possible only with the advent of the telegraph, which made it possible to collect weather data from many places at once. Today the internet makes it possible not only to collect data from many sources, but also for people to collaborate simultaneously. Two projects use the Internet to allow anyone to contribute to important scientific research. One, the Stardust@home project (http://stardustathome.ssl.berkeley.edu/), allows users to help scan a collector that flew through a comet for star dust particles. Similarly, the GalaxyZoo project (http://galaxyzoo.org/) trains users to recognize different types of galaxies so that they can classify a million galaxies. These projects allow not just amateurs, but novices to contribute materially to the advancement of science.

And so it is with software development. The hardware necessary to develop software is ubiquitous. The software most often used to develop software is free. The result is a legion of programmers willing to work for free.

Computer Use in High Schools

It is not yet clear that using computers in classrooms results in increased learning. One study looked at high schools in Silicon Valley with the assumption that if *any* schools were made better by computers, they would be in this tech-savvy area. The examples of profound changes in habits of teaching and learning were minimal (Cuban, 2001). More recently, a study designed to determine whether math and reading software resulted in increased learning similarly found little evidence of increased learning (Dynarski, et al, 2007).

What each of these studies has in common is that students have very little access to computers. Most of the classrooms in Cuban's (2001) studies had only a single computer. In the Dynarski, et al study, students used the reading or math software less than 10% of instructional time. With so little access to computers it should not be surprising that there is little change. When there are sufficient numbers of computers for them to be integrated into classroom use, however, the results are quite different. In a large-scale national study, Becker (2000) found that teachers with a constructivist teaching philosophy, average computer skills, and five or more computers in their classrooms often use computers in their classrooms. But even here, Becker's definition of "frequent" was merely twenty times a year.

The difficulty in showing that computers are beneficial has not been limited to education. For decades, economists have had similar difficulty showing that computer use resulted in increased productivity (Baily and Gordan, 1988). One explanation is that the benefits of new technologies do not happen immediately, they must be ubiquitous for long enough for processes to be redesigned to take advantage of the new affordability of the technology. Electric motors, for example, did not immediately make manufacturing more efficient until factories and

production processes were redesigned (David, 1990). It is reasonable to expect that large-scale measurable changes in how computers affect schools are yet to come. Getting enough computers into schools that they become a regular part of every students' learning will be the first step.

Free/Open Source Software for Science Classrooms

FOSS offers many possibilities for high school science classrooms. One possibility that offers the most opportunity, but requires the most buy-in from someone with technical skills, is to use *only* Free software. Another is to continue to use the operating system currently in place and use FOSS applications. Another option is merely to add some software designed especially for science educators.

How Free Software Can Make Free Computers

Stallman's dream was to use no proprietary software. Though he might argue the benefits of using only FOSS from a moral and philosophical perspective, there are practical and pragmatic reasons as well. For example, Edubuntu and the K12 Linux Terminal Server Project (K12LTSP) are both complete packages including all the applications one needs to use a computer in a classroom. Though both can be used on stand-alone systems, they are also able to convert one computer into a terminal server that can power several *thin clients*. *Thin clients* are responsible only for displaying programs and taking input from the keyboard and mouse; the applications themselves run on the server. Since computers spend most of their time waiting for the next request from the user, a typical teacher's desktop machine is capable of running 5-10 *thin clients* with performance similar to or faster than the same computer running Microsoft Windows. The *thin clients* can be machines that would otherwise be disposed of and need not have hard drives. Because all software resides on a single machine, maintaining thirty machines in a high school classroom requires little more work than maintaining one.

A study of *thin client* systems in a school district found them to be much easier to maintain. A single technician could manage 2,000 systems, whereas it took three technicians to manage 1,500 traditional workstations. Additionally, teachers using the *thin clients* were much better able to integrate computer use into their instruction (Sandholtz, 2004).

Another study looked at providing three high school teachers with enough Linux *thin clients* to give their classrooms near 1:1 student-to-computer ratios. In spite of providing virtually no training in using the computers or integrating them into their classrooms, all of these teachers made significant changes in how and how often their students used computers. In particular, a teacher who had never before used computers himself wholeheartedly embraced the computers and changed how he teaches. Among the benefits that contributed to the systems' success was that every student had his or her own file space and program settings, like web bookmarks, which were available from session to session at any computer they used (Pfaffman, 2007).

Productivity Applications

The most-used applications in and out of the classroom are those in the traditional office suite: the word processor, spreadsheet and presentation tool. Microsoft's offerings are so well-established that Word, Excel and Powerpoint are eponyms, such that many people are unaware that these each of these programs is merely an instantiation of a particular class of programs. OpenOffice.org is a similar office suite that provides these capabilities, will read and write Microsoft Office documents and provides a more familiar interface than Microsoft's Office 12 (Computerworld, 2005). Similarly, The GNU Image Manipulation program (The GIMP) offers a similar feature set to Adobe Photoshop; for those familiar with Photoshop, GIMPShop is a version of the GIMP that mimics Photoshop's menu structure. NVU provides enough web editing features such that most science teachers can do without Dreamweaver or Frontpage. PDFCreator allows any Windows application that is capable of printing to create a PDF file. These applications meet the needs of most science educators. Moreover, being able to install them on any teacher's or student's com-

puter without worrying about licenses is very convenient. A study of open source software use in middle and high schools in Britain found that the open source applications were perceived to be easier and simpler to use than the proprietary equivalents (British Educational Communications and Technology Agency, 2005).

Software Specifically for the Science Teacher

The science community, in its sense of openness, produces a wide variety of software that is freely available. Though much of it is designed by scientists for scientists, there are many applications that are appropriate for use in K-12 classrooms. Another source of software for science classrooms is the science education community.

Running Edubuntu or K12LTSP also provides access to some science applications not currently available for Microsoft Windows or the Macintosh. One program not available for Microsoft Windows or the Macintosh is Kalzium, an atomic table application that contains atomic data like weight, energies, and date of discovery. Kalzium also includes pictures of each element and has a molecular weight calculator. Being able to quickly view the atomic model of many elements makes it easier to understand the power of the periodic table.

The OpenScience Project is another example of a group of scientists working to develop and track tools useful to scientists. They have developed two tools, JMOL and Jchempaint, 3-D and 2-D molecule modeling tools, respectively. JMOL includes an applet that allows students to rotate molecules in a web browser, so students can view molecules without taking time to install software.

Physics Education Technology (PhET) is a set of about 50 simulations created by a group at the University of Colorado (Perkins, et al, 2006). PhET tools are designed to help students make connections between real-life phenomena and science. These simulations are written in Java or Flash so that they run in a web browser without having to install software on local computers. Schools with slow or unreliable internet connections can instead install PhET on local hard drives.

The large and active community of amateur astronomers has produced several FOSS applications for studying astronomy. Stellarium is a powerful planetarium that not only can show a realistic 3-D sky emulating what one might see with a telescope or binoculars, but also can overlay images of constellations. It includes the planets and their satellites in our solar system and images of thousands of astronomical objects. Celestia, unlike most planetarium software, allows you to change your point of view to view the solar system from other vantage points. Kstars, included with Edubuntu and K12LTSP, is another desktop planetarium package that can control a variety of telescopes. Introducing students to these free applications at school empowers them to continue their study of astronomy at home.

Finally, concept maps are increasingly popular in science classrooms as a means of assessing student learning as well as to help students better understand their own knowledge. Freemind is a FOSS concept mapping tool that offers many of the features of Inspiration.

Server-Based Applications

Moodle is a learning management system. Like Blackboard, a widely-used proprietary example, Moodle enables a teacher to publish assignments, a calendar, and course materials with minimal training. It can also allow students to turn in their assignments electronically and to take online assessments. It can help teachers who teach the same course to coordinate their curricula more easily (Perkins and Pfaffman, 2006). Another server-based application is Science Fair in a Box, a web-based system to manage participants, judges, sponsors and awards for a region hosting a science fair.

Conclusion

Though many educators are unaware or dismissive of Free/Open Source Software, the number of FOSS tools that support education in high schools in general and science education in particular continues to grow. As such, the importance and application of FOSS can no longer be dismissed as a mere anomaly. The need to inte-

grate FOSS into science classrooms is essential. Using FOSS offers a practical means to give teachers and students the tools they need in the science classroom. The ability to use these tools also teaches students about critically assessing the implication of the software tools they use. Finally, FOSS itself models the way that science is developed and propagated, giving students a better understanding for how science works.

References

Baily, N., Martin, & Gordan, R. J. (1988). The productivity slowdown, measurement issues, and the explosion of computer power. *Brookings Papers on Economic Activity, 2*, 347-431.

Beatty, J. K. (2003). Testbed Paves Way for Amateur Space Telescope. *Skytonight.com*. Retrieved March 12 from http://skytonight.com/news/3305551.html

Becker, H. J. (2000). Findings from the teaching, learning, and computing survey: Is Larry Cuban right? *Educational Policy Analysis Archives, 8* (51).

British Educational Communications and Technology Agency (BECTA). (2005). *Open source software in schools: A study of the spectrum of use and related ICT infrastructure costs*. Coventry, UK: BECTA. Retrieved from http://publications.becta.org.uk/display.cfm?resID=25907 January 14, 2008.

Considering changing your school's computer system to open source software? one school's conversion to Linux has been a complete success. (2003). *Canada's SchoolNet*.

Cuban, L. (2001). *Oversold and underused: Computers in the classroom*. Cambridge: Harvard University Press.

David, P. A. (1990). The dynamo and the computer: An historical perspective on the modern productivity paradox. The American Economic Review, 80 (2), 355-361. (*Papers and Proceedings of the Hundred and Second Annual Meeting of the American Economic Association*)

Dynarski, M., Agodini, R., Heaviside, S., Novak, T., Carey, N., Campuzano, L., et al. (2007). *Effectiveness of reading and mathematics software products: Findings from the first student cohort* (Tech. Rep.). Washington, D.C.: U.S. Department of Education, Institute of Education Sciences.

Ferris, T. (2002). *Seeing in the dark: How backyard stargazers are probing deep space and guarding earth from interplanetary peril*. New York: Simon & Schuster.

Goldman, R., & Gabriel, R. P. (2005). *Innovation happens elsewhere: Open source as business strategy*. San Francisco, CA: Morgan Kaufmann.

Hepburn, G. (2005). Open source software and schools: New opportunities and directions. *Canadian Journal of Learning and Technology, 31* (1).

Kogut, B. and Metiu, A., (2001). Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy, 17*(2), 248-264.

Latour, B. (1987). *Science in action: How to follow engineers and scientists through society*. Open University Press, Milton Keynes.

Migrating to OpenOffice.org 90 per cent cheaper than to Microsoft Office 12. 2005. *Computerworld, 11*(23). Retrieved February 8, 2007 from http://computerworld.com.sg/ShowPage.aspx?pagetype=2&articleid=2742&pub%id=3&issueid=66

Netscape announces plans to make next-generation communicator source code available free on the net (1998, January). Retrieved October 30, 2007 from http://wp.netscape.com/newsref/pr/newsrelease558.html

Perkins, K., Adams, W., Dubson, M., Finkelstein, N., Reid, S., Wieman, C. .2006. Phet: Interactive simulations for teaching and learning physics. *The Physics Teacher, 44*, 18.

Perkins, M. and Pfaffman, J. (2006). Using a course management system to improve classroom communication. *The Science Teacher, 73*(7), 33-37.

Pfaffman (2007). Transforming Instruction without Training: A Case Study of the K12 Linux Terminal Server Project. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007*. 363-366.

Raymond, E. S. (1997, Nov). *The cathedral and the bazaar*. [Online.] Available: http://www.catb.org/~esr/writings/cathedral-bazaar/

Sandholtz, J., and Reilly, Brian (2004). Teachers, Not Technicians: Rethinking Technical Expectations for Teachers. *Teachers College Record, 106*(3), 487-512.

Stallman, R. (1985), 03 March. **The GNU manifesto.** Retrieved Mar 12, 2007 from http://www.gnu.org/gnu/manifesto.html

Williams, S. (2002). *Free as in freedom: Richard Stallman's crusade for free software*. Sebastopol, CA: O'Reilly.