# OPEN SOURCE SOFTWARE

# AND COMPUTER SCIENCE EDUCATION

*Keith J. O'Hara and Jennifer S. Kay*
*Computer Science Department*
*Rowan University*
*Glassboro, NJ 08028*
*{ohara, kay}@elvis.rowan.edu*

**ABSTRACT**

We investigate the role of open source software in computer science education. We begin with a brief tutorial on open source software including a description of four popular open source licenses. Next we discuss the use of open source software in education. Finally, we focus on the use of open source software in computer science education.

## INTRODUCTION

As computer science educators we constantly seek new channels, methods, and technologies to reach and intrigue our students. We hope to first capture their interest, then maximize their understanding and retention of the material, and finally encourage their own independent creative work. Throughout this process, we try to teach them skills that they can apply in the real world. The breadth of our field and the variety of pedagogical approaches make this process very difficult.

We believe that open source software (OSS) can serve as a channel, method, and technology to teach and learn computer science. OSS has the potential to expand group work beyond the classroom to include much larger projects and more distributed teams. OSS can also be used to introduce our students to the larger computer science community and to the practice of peer-review. Finally, OSS can often provide us with free or lower-cost technology in the classroom, permitting us to use technology that we might otherwise be unable to afford.

In this paper, we discuss the role of open source software in computer science education. First, we give a brief tutorial on open source software including a description of four popular open source licenses. Next we discuss the use of open source software in education. Finally, we focus on the use of open source software in computer science education.

## OPEN SOURCE SOFTWARE

The Open Source Initiative's definition of open source software is: "software that must be distributed under a license that guarantees the right to read, redistribute, modify, and use the software freely." [14] As much a part of the Open Source Software (OSS) development model as the open source code is a global network of developers. The software goes through a type of natural evolution - resulting in rapid development, increased reliability, and decreased cost.

Open source software is also referred to as "free software". Although the two are interchangeable under the Open Source Initiative's definition, there are some differences among popular understanding. When marketing the "open source" approach we often rely on the pragmatic properties of the open source code. For instance, we may cite the software's reliability and high quality due to the peer review and rapid evolution of the source code. [14]

The disparities lie in exactly who gets access to this source code and why they have access to it. When explaining free software we use a moral argument; rather than basing our argument on the benefits of the open source code, we purport the *right* everyone has to the source code. When we speak of "free software" we are referring to the freedom of the user to read, redistribute, modify, and use the source code rather than the lack of cost.

## OPEN SOURCE LICENSING

Open source software can be released under a variety of different licenses. As of February 2002, the Open Source Initiative has approved 30 open source software licenses. [13] We will discuss four licenses common to educational applications: the GNU Public License, the Lesser GNU Public License, the Berkeley Software Distribution License, and the Mozilla Public License.

- **GNU GPL.** The GNU (GNU's Not Unix) General Public License (GPL) is a very common open source license. This license assures any derived work of a GPL program will remain GPL, thus assuring the freedom for future users/developers to access, modify, and redistribute the source code. The software under this license is *copylefted*, "copylefted software is free software whose distribution terms do not let redistributors add any additional restrictions when they redistribute or modify the software."[10] The copyleft mechanism provides "incentive" for developing and improving free software. [8]

  Popular software: Linux Kernel, GNU Emacs, and GCC.

  Percentage of software listed on freshmeat.net[1]: 65.15%

- **GNU LGPL**. The Lesser GNU General Public License, as the name implies, is not as free as its older brother. The LGPL allows proprietary code to be linked with the LGPL code. For instance, if you were writing a library and were to release the source under the LGPL, code under a different license would be permitted to link with your library.[2] The software under this license, like the

[1] "Freshmeat maintains the Web's largest index of Unix and cross-platform software."[7]

[2] In fact, the 'L' initially stood for "Library" but was later changed to stand for "Lesser".[6]

GPL, is also copylefted, but unlike the GPL, can be linked to proprietary code with its own license.

Popular software: GNU C Library, OpenOffice, and Fast Light Toolkit (FLTK)

Percentage of software listed on freshmeat.net: 5.31%

- **BSD**. The Berkeley Software Distribution License is a simple non-copylefted license. This license allows derived works to be released under any license. This allows BSD code to be used in proprietary projects and has no copyleft mechanism for keeping any of the modified code open. For this reason, the license is criticized because it does not offer any "incentive" for developers because there is no guarantee the code will remain open.

  Some popular software: Apache, FreeBSD, and PostgreSQL

  Percentage of software listed on freshmeat.net: 5.17%

- **MPL**. The Mozilla Public License (MPL) permits the use of the code in proprietary projects like the BSD License, but makes certain the core files stay under the MPL. This stipulation, that the core files stay under the MPL, provides incentive for developers to improve and develop aspects of the core functionality. This license is actually GPL-incompatible; a GPL module cannot be legally linked with a MPL module. [9] "However, MPL 1.1 has a provision (section 13) that allows a program (or parts of it) to offer a choice of another license as well. If part of a program allows the GNU GPL as an alternate choice, or any other GPL-compatible license as an alternate choice, that part of the program has a GPL-compatible license." [9]

  Popular software: Mozilla Web Browser and Bugzilla

  Percentage of software listed on freshmeat.net: 0.49%

So how does one pick a license for an educational application? David Baum, the creator of Not Quite C (NQC), a programming alternative for the LEGO MindStorms Robotics Invention System, chose the Mozilla Public License (MPL), but why?

> "The biggest difference between MPL and GPL is that the GPL is more stringent about keeping things even partially derived from GPL code to still be GPL, whereas MPL allows MPL code to be combined with proprietary code to result in a proprietary project. From the standpoint of social engineering, the GPL does a better job promoting free software. However, it also means that in some cases GPL code is not as widely accepted as other open software licenses. My primary motivation for releasing the NQC source was to allow as many people as possible to use it, and I wasn't concerned about the chance that others could profit from my work, so MPL seemed to fit better." [1]

The GPL and LGPL assure the code will stay open, but they raise issues about widespread acceptance, especially among BSD-minded developers and the corporate community. The BSD license permits the code to be used in closed projects, but does not have the "incentive" for developers to recontribute fixes or additions. This incentive has been pointed to as the reason why GNU/Linux has formed a stronger community than the BSDs. The MPL tries to maximize both widespread acceptance and GPL-like development, hoping to strike a balance between the "free" and the

"proprietary". Regardless of your license, by releasing your code under an open source license you allow others to find bugs, improve the algorithms, write documentation, or port the software to other hardware and software configurations.

## OPEN SOURCE SOFTWARE AND EDUCATION

Many groups have been established to develop and promote open source software as a viable technological solution for educational uses. By using open source software, schools can free themselves of licensing costs and put their (often scarce) resources to other uses. As one might expect, the focus is on Linux, and tools that work on Linux platforms.

Red Hat, a company best known for its commercial support of GNU/Linux, has developed the *Open Source Now* project. [15] Open Source Now is an advocacy group designed to advance the use of OSS in both education and public policy. For educators, they provide an introduction to OSS in education, a discussion list, and links to a wide variety of other OSS in education sites.

*Simple End User Linux - Education* is a discussion list about all aspects of educational uses of Linux. [16] The site also provides a database of case studies, and links to current projects and a wide variety of software from astronomy to social studies.

The *K-12 Linux Project* provides information to schools to enable them to make use of OSS. [11] For example, their Linux Terminal Server project has produced an open source terminal server Linux distribution targeted at a lab of low-powered diskless computers and a high-powered server. This allows a school to take advantage of outdated hardware they may already own, as well as save money by reducing the software licenses the schools must purchase. They claim to cut a lab's hardware and software costs by more than a third.

Open source software is not limited to low-cost computer labs in primary and secondary schools, but also is being actively used and developed by higher education institutions. The *OpenScience* project is dedicated to developing scientific software that can be taken advantage of freely by anyone. [12]

Relying on the flexibility, low-cost, standards compliance, and stability of OSS to strengthen its credibility as an educational solution, we must also recognize the factors that inhibit its widespread acceptance. Today's schools are very *WinTel*[3] reliant, possibly due to historical vested interests, a steep learning curve, or masterful marketing. Another factor, which is sure to surface as another inhibitor, is the current mucky situation of open source licensing. While supplying a great amount of flexibility for developers, this plethora of licenses has a steep learning curve of its own. It also breeds confusion and uncertainty for users and neophyte developers.

## OPEN SOURCE SOFTWARE AND COMPUTER SCIENCE EDUCATION

Open source software offers tremendous benefits to the computer science education community. By using OSS, computer science educators and students can develop (or further the development of existing) software that can be used and improved upon by an international community. This not only provides the student with a world-size laboratory and support staff, but also gives them experience in large-

[3] Combination of Microsoft Windows and Intel Processors

scale software collaboration and development. Distributed software collaboration has proven itself effective in the educational setting through the use of Internet-driven collaboration tools such as web pages and email lists. [3] These Internet-driven collaboration tools are the same as those that power open source development. Thus, we can look to the open source community to expose our students to large-scale distributed software development. By using and developing OSS, not only is the student participating in a large distributed software community, but is also interacting with large, *real*, software code-bases.

GNU/Linux, an open source operating system, has also been presented as an educational technology to teach networking, databases, and system administration at the university level. [5] The use of OSS is particularly appropriate in a Senior Thesis or other capstone course. Students who are encouraged to build projects on top of OSS bases can build more interesting and exciting systems than they might have developed from scratch. Further, in order to succeed, they must work with a large code-base and associated documentation. If the code is well written and well documented, students learn how much can be accomplished with careful preparation. If it is not, students may take more care with their own work in the future. The best of these systems can be contributed back to the open source community.

Finally, due to the open and freely available source code, a high degree of verifiability can be attained. [2] Just as we look to conferences and journals for peer-review of our ideas, we can look to the open source community for the same type of scrutiny concerning our software. [4] By taking advantage of the open source development model we extend the methodology by which we learn, apply, and teach computer science, to include peer-review. We may look away from the pragmatic properties of the open source code to the more “moral” argument a free software advocate may pose. Free software assures everyone will have access to your code, which as a computer scientist, are your ideas. As a scientist it is your duty to assure *everyone* can understand, modify, and apply those ideas.

## CONCLUSIONS

The use of open source software, and GNU/Linux in particular, is growing in all directions. Ranging in application from the embedded world to corporate infrastructure, open source software also gives educators a type of flexibility and intellectual freedom often absent from software. Open source software can serve as a channel, method, and technology to teach and learn computer science. As a channel, OSS can expand teamwork past the classroom to include much larger projects and more distributed teams. As a method, OSS can be used to introduce our students to the larger computer science community and to the practice of peer- review. Finally, acting as a technology, OSS can provide us with free or lower-cost technology in the classroom that we might otherwise be unable to afford.

## REFERENCES

[1] Baum, D. Email interview. October 11 2001.

[2] Kiernan, V. “The ‘Open Source Movement’ Turns Its Eye to Science.” Chronicle of Higher Education. November 5, 1999. http://www.chronicle.com/free/v46/i11/11a05101.htm [accessed January 2002]

[3] Macek, T., Mannova, B., Kolar, J., and Williams, B. Global Cooperation Project in Computer Programming Course. In Proceedings of SIGCSE Annual Conference (1999).

[4] Pfaffenberger, B. “Linux in Higher Education: Open Source, Open Minds, Social Justice.” Linux Journal. March 21, 2000.linuxjournal.com/article.php?sid=5071 [accessed February 2002]

[5] Rogers, M. Working Linux into the CS Curriculum. In Proceedings of the Seventh Annual CCSC Midwestern Conference (2000).

[6] Stallman, R. “Why you shouldn’t use the Library GPL for your next library.” Feb 1999. http://www.gnu.org/philosophy/why-not-lgpl.html. [accessed February 2002]

[7] About. Freshmeat.net http://freshmeat.net/about/ [accessed February 2002]

[8] What Is Copyleft? http://www.gnu.org/copyleft/copyleft.html [accessed February 2002

[9] GNU License List. http://www.gnu.org/licenses/license-list.html [accessed February 2002]

[10] GNU Philosophy. http://www.gnu.org/philosophy/ [accessed February 2002]

[11] K-12 Linux Terminal Server Project. http://www.k12ltsp.org/press.html [accessed February 2002]

[12] The OpenScience Project. http://www.openscience.org [accessed February 2002]

[13] Open Source Initiative. http://www.opensouce.org/ [accessed February 2002]

[14] Open Source Initiative. Frequently Asked Questions. http://www.opensouce.org/advocacy/faq.html [accessed February 2002]

[15] Open Source NOW. http://www.redhat.com/opensourcenow/ [accessed February 2002]

[16] Simple End User Linux - Education. http://www.seul.org/edu/ [accessed February 2002]