Open-Source Projects

Scenario
Halfway through the semester, Dr. Margaret Broadwater was excited by the progress her students were making in her course Open-Source Software Development. Working with open-source software projects and development communities gave her students hands-on experience with software development practices, technology frameworks, data structures, and product development. Students also completed installation exercises for open-source projects from both developers’ and users’ perspectives, followed by finding and patching bugs in the software.

Broadwater knew that her students were learning more than just how to work their way around code. In talking with students she emphasized that open-source code was the heart of applications that had become ubiquitous in business and education, including Chrome and Firefox, and was the driver for software like the Apache web server, Fedora Linux, and OpenSSL. Moreover, open source had gained purchase in use by companies, organizations, and government agencies and was thus something they would need to know once they entered the workplace as software developers and engineers. Broadwater knew that by working on open-source projects in depth, her students were also learning about the ethos of building code in a community of developers—and, indeed, were becoming part of that community.

Broadwater had learned about open source in a summer workshop sponsored by POSSE, the Professor’s Open Source Software Experience. That meeting focused on helping instructors learn to engage students in development of free and open source software (FOSS). Broadwater learned about FOSS tools and the FOSS culture, picked up a cache of rich curricular materials, and learned pedagogical techniques for supporting student engagement in learning about open source. Another plus was that the instructor for the workshop volunteered to serve as a mentor for Broadwater’s class. Working occasionally with students via BigBlueButton, an open-source web conferencing system, he helped steer learners through assignments and answered their most challenging technical questions.

By the end of the semester, Broadwater’s students had successfully learned many of the intricacies and idiosyncrasies of open-source projects. Some were still learning particulars, like fixing bugs. Other students were eager to move beyond the fundamentals, and Broadwater urged them to focus on suggesting specific improvements or additions. Several students made that assignment the focus of their senior projects.

1 What is it?
Open source refers to software that can be freely used, changed, and shared. Open-source projects engage a community of developers who collaborate and iterate to develop, grow, and improve software. Developers can access, modify, and customize functionality, providing considerable control to organizations that implement open-source software. In general, open-source projects seek less expensive and more flexible paths to solutions that are not being adequately or affordably addressed by proprietary software. Typically, communities that initiate an open-source project establish a set of values and governing principles concerning factors such as priorities for features, technical specifications, workflows, and how the community working on the project will make decisions. Organizations that build open-source software for education include Aperseo, DuraSpace, Kuali, and Moodle. The nonprofit Open Source Initiative develops standards and certifies licenses for open-source code.

2 How does it work?
Colleges and universities engage in open-source projects in a variety of ways. One impetus for considering open-source projects is when a campus believes it can better meet its software needs in-house or by collaborating with peers from other institutions. Vendor solutions might not address the problem adequately or even at all, or they might be too expensive or too restrictive in terms of licensing. In an open-source community, a “coalition of the eager” might define a problem and sketch a framework for a solution. Project leaders would then frame specific parameters for the project, how the work should be governed, how the workload might be shared, how costs would be divided, what risks might be involved and how they would be mitigated, and how the project could best be maintained over time. In some cases, an individual institution will adopt an open-source solution and then join the development community by submitting issues, fixing software bugs, and suggesting ideas for development. As they become more engaged in the community, institutions may contribute code to existing features or perhaps develop a new tool or module. Open-source projects are usually emergent in nature rather than planned and typically grow organically to address a given need.
Open-Source Projects

3 Who’s doing it?
Open-source projects vary greatly in scale. Some are driven by individuals creating a specific application or “widget.” Others are major initiatives developed by institutions, companies, and consortia. Colleges and universities have long used open-source code in applications such as Apache web servers and CAS authentication. Other examples frequently found in higher education include Linux, Drupal, Wordpress, MySQL, Postgresql, and Mozilla. OpenCast, Tsugi, and uPortal offer open-source solutions for teaching and learning, while platforms such as Moodle, Kuali, and Sakai provide solutions for course management and financial systems. Companies including Facebook, Google, IBM, HP, Microsoft, and Oracle often incorporate open-source code into solutions they sell. Some open-source development takes place under the aegis of well-established consortia or communities, such as the Apereo Foundation, DuraSpace, and Teaching Open Source.

4 Why is it significant?
In contrast to proprietary products, open-source solutions typically offer more control over features, functionality, maintenance, and costs, with easier pathways to address security vulnerabilities, software bugs, and evolving needs, often using agile development methodologies. Users of open-source code can develop software that fits their needs and participate in defining what the solution is and does, rather than trying to adapt to specifications designed by third parties. Open source offers innovative, customizable, cutting-edge solutions with freely visible standards and specifications. Open-source code is free to purchase and often can be accessed without burdensome procurement processes. Open source does not tie users to vendor constraints or mandated upgrades. Maintenance can be done in-house, collaboratively with other community members, or with service providers—including cloud providers—on an institution’s schedule. Development time can be faster than for commercial solutions. Transparency in development helps users evaluate code for efficacy and compatibility and discover problems that can be corrected without waiting for a vendor to act. Open-source implementation can readily act as a reference for others while proprietary code cannot.

5 What are the downsides?
Open-source projects incur the costs associated with using any software, such as maintaining required hardware and network infrastructure, server/database configuration, general management and maintenance, user support, and staffing and training. If the institution elects to participate in development, it will incur additional costs. Although the ability to alter open-source software is a key benefit, sometimes developers disagree or have different goals, which can lead to a split (or a “fork”) in the codebase, resulting in separate, sometimes competing software programs. In addition, some kinds of software offer numerous open-source options, while others have none.

6 Where is it going?
Open source is widely used, and its impact is likely to grow. Economic pressures and the expanding scope of services are likely to drive more universities to pursue open-source options. Increasingly, too, institutions will find that commercial partners use open-source code in certain applications. Colleges and universities will become more comfortable with open-source solutions for core functions such as financial management, enterprise resource planning, course and program administration, grants management, and teaching and learning. While practices for standardization and licensing for open source are now well-accepted, questions might arise about license compatibility or patenting around specific licenses.

7 What are the implications for teaching and learning?
Open source gives higher education more control over its application portfolio and can contribute to the next generation digital learning environment. Solutions such as Apereo OAE, CAS, or the Tsugi project have the potential to transform learning management and student support services and enhance student success. Open-source code is helping expand access to affordable education, and solutions such as Moodle and Sakai are helping advance personalized learning. Opportunities to engage in the development of open-source code give students practical experience and practice in project-based management models. Open-source projects help faculty and researchers share their work and develop ideas broadly with colleagues around the world and offer new ways for libraries to access and archive digital collections.