innovate
journal of online education
OCTOBER/NOVEMBER 2006
VOLUME 3, ISSUE 1

The *Innovate* Gateway
October/November 2006
Volume 3, Issue 1

Welcome to the October/November issue of *Innovate*. Among the range of technological developments that hold significance for educators, one of the most far-reaching in its future implications is the rapid growth of open source software. In this special issue of *Innovate*, our contributors explore, assess, and illustrate the potential of open source software and related trends to transform educational practice.

Our first four articles map out the current state of open source technology and offer recommendations for how educational institutions can benefit from its advances. **David Wiley** sets the stage by offering a recent history of the open source movement and discussing its recent impact in the educational sector. As Wiley notes, the emergence of open source applications for instructors and administrators has been accompanied by other significant trends such as open course content and open access publishing venues for academic research. In assessing these developments, he argues that the "culture of openness" they foster will allow educational institutions to maintain the flexibility, transparency, and collaborative spirit they need to serve future generations. In turn, **Robert Stephenson** argues that the community networks established by open source software initiatives provide a valuable model for similar networks in the educational sphere. In his commentary Stephenson outlines his concept of open course communities, a "knowledge ecosystem" in which the development and assessment of course materials would arise from technology-enhanced grassroots collaboration among educators, designers, librarians, and students themselves.

Meanwhile, for many institutions the actual adoption of open source software still remains an "open" question; focused advocacy and strategic foresight thus remain the watchwords in our next two articles. In their commentary **Gary Hepburn** and **Jan Buley** first describe the implementation strategies available to schools considering open source software, and they subsequently address the key sociopolitical factors that must be taken into account by advocates of such implementation. **Patrick Carey** and **Bernard Gleason** note that open source software has resulted in significant advances in commercial software as well, which has led to the possibility of adopting modular combinations of open code and proprietary applications. In order to take full advantage of these trends, they argue, institutional planners should ensure that their systems provide an open, standards-based architecture that allows for a flexible range of software options.

Our following contributors offer detailed accounts of the development, design, and use of specific open source applications as well as a study of how the process of open source development provides a valuable model of pedagogical design in its own right. **Toru Iiyoshi**, **Cheryl Richardson**, and **Owen McGrath** introduce readers to the KEEP Toolkit, a set of software tools designed to provide graphic representations of teaching practice and thereby support focused inquiry into pedagogical strategies. After describing the features of this application, they illustrate its early use by educators and then outline the stages whereby they eventually released it as open source software. **Harvey Quamen**

illustrates how he used MySQL software and PHP code to create a database that would streamline editorial tasks and procedures for a journal on humanities research; based on the early results of this project, he advocates the use of these tools for academic journals with limited financial resources. **Kun Huang**, **Yifei Dong**, and **Xun Ge** propose that the collaborative work environment of open source development has a distinctively pedagogical value for computer science instructors. In illustrating this claim, they describe a graduate computing course in which student teams worked on software design projects in an online environment modeled after the virtual workspaces of open source software initiatives.

Finally, in his Places to Go column, **Stephen Downes** introduces readers to Intute, an open access Web site that represents a significant step forward in the evolution of learning object repositories. Through the distinctive design of its search feature, Intute exposes its users to a much broader network of resource providers than typically provided by other repositories. With its plans to release its own software as open source, Intute also promises to spur the growth of similar repositories that will further fuel vital innovations in teaching practice.

We hope you enjoy this issue of *Innovate*. As always, we look forward to interacting with you through the journal's discussion boards, live webcasts, and other exemplary features.

Vijay Kumar
Special Issue Editor

James L. Morrison
Editor-in-Chief

Open Source, Openness, and Higher Education

by David Wiley

With the growth of open source software and other related trends, a culture of openness is advancing from the edges of society to the core of academic culture. In this article I provide an overview of how the expansion of open source software in culture at large has affected the world of education, describe how the greater use of open source software in education has unfolded hand-in-hand with the development of open course content and open access research, and argue that this more comprehensive shift towards "openness" in academic practice is not only a positive trend, but a necessary one in order to ensure transparency, collaboration, and continued innovation in the academy.

From the Margins to the Mainstream: A Recent History of Open Source Software

There once was a time when open source software was the sole province of the geek and existed behind barriacades impassable by ordinary computer users. The first major barrier was inscrutable jargon; users who did not understand the meaning of commands such as "./configure; make; sudo make install" were often simply left out. A second obstacle was that most open source programs, such as Web servers and mail servers, were limited primarily to applications and utilities that were useful to servers and network administrators; such resources went beyond the needs of the average desktop computer user. Even those applications that might have broader appeal, such as text editors, were often so complex that running them seemed to require a specialized degree. A further limitation was that most open source software was written exclusively for free, Unix-like operating systems. Finally, these tendencies, in turn, contributed to another major barrier—namely, the common perception among average computer users that free applications must surely lack the intuitive features and accompanying technical support of commercial applications. In short, the world of open source was closed off to normal people.

Recent developments, however, are bringing open source into the lives of average desktop computer users. In addition to their availability for open source operating systems like Linux, many open source applications are also available for proprietary operating systems like Windows and Mac OS X. Installing these applications generally works like installing commercial software—users just double-click to begin the install process. In many ways, recent trends in the open source world can be seen as a distributed effort to replace popular proprietary software with easy-to-install-and-use open source software. Options for nearly all commonly used programs abound. For example, in place of Microsoft Office, users can run Open Office, an MS Office-compatible open source replacement. One alternative to Microsoft's Internet Explorer is Firefox, an open source Web browser with all the features of its commercial counterpart. Instead of Microsoft Outlook, users can try Thunderbird, a full-featured, open source e-mail application. Jabber not only offers an open source alternative to AOL's Instant Messenger but also allows users to chat with people running software from AOL, MSN, Yahoo!, and ICQ. Word processing, working with spreadsheets, crafting presentations, surfing the Web, managing e-mail, and chatting with co-workers and friends probably account for the vast majority of the time average desktop computer users spend at their machines. While these users and their needs for simple installs and familiar interfaces were all but ignored only a few years ago, today it is actually possible for anyone to perform the basic tasks outlined above without ever launching a proprietary application. Even more specialized needs can be satisfied with alternative sources of software for users who look for them. For example, one can visit Google, enter "open source photoshop" or "open source mathematica," and discover programs such as GIMP and Maxima, respectively, or a host of other versions of completely free and open source software.

Open source software enters mainstream use when it compares favorably to proprietary applications in terms of available features and technical support. In some cases, freely available software surpasses commercial rivals in these critical areas and thus compels the industry to improve its products. To cite just two examples, Firefox users for years have taken for granted certain features such as tabbed browsing that are only now being adopted into a beta version of Internet Explorer, and users of Apache have enjoyed excellent e-mail and chat support. In these ways the open source movement has helped raise standards within the market at large and has served as a catalyst for productive innovations that many people now regard as essential to software functionality and design.

The Empowering Nature of Open Source Software

In today's economy, the ability to provide services or create products that people value is generally impossible without computing and software capabilities. When the cost to purchase or license software is prohibitively expensive, all but the very wealthy are prevented from innovating in the areas of new products and services. Much like the exploited workers of Marx's writings, who cannot afford expensive machinery and so must eternally rent their labor to someone else already rich enough to afford factory machines, normal computer users find themselves unable to afford the high price of much proprietary software, and this prolongs the situation in which workers cannot afford to own the means of production or direct their own labor.

Such a rigidly hierarchical set of relations has now begun to change. Open source software empowers users by making a critical part of the means of production—software—freely available, allowing ordinary people to direct their efforts as they please. The recent history of the Internet is, in fact, the history of tens of thousands of people empowered to bring a variety of innovative services and products to market because the tools for creating them were within their financial reach. It is a history of average users armed with open source software competing successfully with multinational corporations armed with expensive, proprietary software.

The impact has been substantial. In an April 2006 Netcraft survey (2006) of 81,565,877 Web sites, over 64% of all sites surveyed were running the open source Web server Apache as compared to 25% running Microsoft's IIS Web server. After Web server software, arguably the next most important class of software for creating products and providing services include those tools that are used to write Web-based programs. Open source languages like PHP, Python, Perl, and Ruby proliferate in this area as do open source application development environments like Ruby on Rails, Zope, and PEAR. (Users not familiar with any of these languages or frameworks can access try Ruby from within any browser without installing anything. Anyone can also watch some screencasts of what it is like to develop in Ruby on Rails—I particularly enjoy one application that goes from nothing to a functioning blog in only 15 minutes and 58 lines of code.) This open source, critical infrastructure continues to serve as the fuel driving innovation in practically every sector of the economy.

In addition to using open source software to build Web applications, average users are making increasing use of open source applications for everyday online activities like instant messaging (GAIM), journaling or blogging (WordPress), and sharing pictures with family and friends (Gallery). In this regard the impact of the open source movement has clearly moved beyond the more specialized spheres of software development and Web site management, allowing users at all levels to gain greater access to technological means of production outside the exclusive domain of proprietary software.

Moving into Education

In turn, the growth of open source software in the public at large has begun to make itself felt in various ways within the arena of education. This influence can be seen most directly in the wide array of open source software applications now available to educational institutions and instructors. At the same time, this influence can also be seen in the further steps towards openness that have begun to take place outside the specific context of software use—in particular, the dissemination of open access course materials as well as the creation of open access research repositories and electronic publishing venues.

Software

Educational institutions have a growing array of options to consider when acquiring software. Learning and course management systems (CMS) like WebCT and Blackboard can now be replaced by attractive open source alternatives like Sakai and Moodle. In recognition of these options, Athabasca University (2006) has recently announced their adoption of Moodle as their official CMS, and other institutions such as the British Open University (2005) have taken this step as well. Meanwhile, a plethora of educational applications are available to support student learning in both higher education and K-12 contexts. Open source applications that teach and tutor on every subject are surprisingly abundant—one need only search a site such as SchoolForge to survey the range of options.

Moreover, the potential for the development of education-specific features within existing open source software

promises additional opportunities. Open source applications are, by design, adaptable and can therefore be applied to a variety of uses. For example, photo gallery software designed for sharing and rating photos could be adapted for sharing and rating essays, CAD files, or musical compositions. Software designed for creating interactive multimedia presentations could be adapted for a course module illustrating the phases of bacterial cell growth or the relationship between supply and demand in classical economic theory. If the redesign of non-educational software for educational use has always been possible in principle, the highly flexible design of open source applications provides the ideal way to put this principle into practice.

Teaching

The open source philosophy is having a much broader impact in education than just changing the way universities license software—it is also changing the way faculty disseminate research results and teaching materials. A recent report (Wiley 2006) produced for the Organisation for Economic Co-operation and Development (OECD) makes plain the extent to which educational institutions are adopting the open source mindset in their use of educational resources. Approximately 175 universities worldwide currently participate in programs through which they provide free and open access to the content of over 2,000 university courses; MIT OpenCourseWare provides the majority of these courses, but with efforts underway at universities in Australia, Brazil, Canada, Hungary, India, Iran, Ireland, the Netherlands, Portugal, Russia, South Africa, Spain, Thailand, the UK, the US, and Vietnam, that majority position is not likely to last long. The move to share the content of university courses openly is growing at an amazing rate.

Many other kinds of educational resources are widely shared, thanks to the open content philosophy. Thousands of smaller educational modules are accessible through collaborative online projects such as Connexions. The Web site Textbook Revolution indexes and provides free access to over 150 textbooks made available by their authors and copyright holders. In the spirit of reusing and adapting shared material, many of these resources are being translated into Spanish, Chinese, Portuguese, and other languages. Meanwhile, the Multimedia Educational Resource for Learning and Online Teaching (MERLOT) currently offers access to almost 15,000 educational materials.

Research

For years now an undercurrent of dissatisfaction with traditional publishers of scholarly research has been swelling in academia. The expression of frustration goes something like this: Researchers work hard to develop new research ideas, to attract funding to support the new research, to carry out the research itself, and to describe the results in order to communicate with others, and in exchange for publishing the work produced in universities, journals demand that authors sign over all copyrights to their own research. This leaves researchers in the unfortunate circumstance of not owning the descriptions of the results of their own work. When researchers want to use their own descriptions of their research, they must acquire the agreement of—and sometimes even provide payment to—the journal that owns the copyright to the researcher's work product. The question is repeatedly raised: How did this situation arise, and what can be done about it?

Faculty seeking alternatives should consider projects such as arXiv.org. According to the Web site, arXiv is "an e-print service in the fields of physics, mathematics, non-linear science, computer science, and quantitative biology" that provides open access to over 367,900 academic papers as of May 2006 (arXiv e-Print archive, n.d.). While papers in the archive are not necessarily peer-reviewed, the collection is generally of excellent quality.

Another response to the power of publishers is an initiative called the Public Library of Science (PLoS), which is peer-reviewed. The PLoS Web site recounts the history of its founding:

Following its founding in October 2000 by biomedical scientists Harold E. Varmus, Patrick O. Brown, and Michael B. Eisen, PLoS's first action was to circulate an open letter encouraging scientific publishers to make the research literature available for distribution through free online public archives such as the US National Library of Medicine's PubMed Central. This letter, signed by nearly 34,000 scientists from 180 countries, prompted significant steps by many scientific

publishers towards freer access to published research. Unfortunately, the publishers' responses fell far short of the reasonable policies we advocated.

In 2003, PLoS launched a nonprofit scientific and medical publishing venture that provides scientists and physicians with high-quality, high-profile journals in which to publish their most important work. Under the open access model, PLoS journals are immediately available online, with no charges for access and no restrictions on subsequent redistribution or use, as long as the author(s) and source are cited, as specified by the Creative Commons Attribution License. (n.d., "What We Do," ¶ 1-2)

Importantly, because journal impact factors depend heavily on how many times articles from a given journal are cited, open access journals have a huge advantage over traditional restricted-access journals. *PLoS Biology* has been in existence only since 2003; yet when Thomson ISI calculated the journal's very first impact factor in 2005, it was 13.9, making *PLoS Biology* the most-often cited journal in general biology, ahead of more established outlets like *Proceedings of the National Academy of Sciences*. I am unaware of another research journal that has, so to speak, debuted at number one. In a recent article in *PLoS Biology*, Eysenbach (2006) describes the benefits of OA (open access) publication:

In a logistic regression model, controlling for potential confounders, OA articles compared to non-OA articles remained twice as likely to be cited (odds ratio = 2.1 [1.5-2.9]) in the first four to ten months after publication (April 2005), with the odds ratio increasing to 2.9 (1.5-5.5) ten to sixteen months after publication (October 2005). (¶ 1)

Eysenbach modestly concludes that "OA is likely to benefit science by accelerating dissemination and uptake of research findings" (¶ 1). Yet as a model for how open access journals can overcome the constraints that researchers face in the field of academic publishing, *PLoS Biology* clearly has broader implications outside the specific field of science. As other disciplines recognize the value of peer-reviewed e-journals for their own research, the precedent set by *PLoS Biology* will help stimulate similar efforts to make open access the norm rather than the exception for academic publication.

Does Open Access Threaten Faculty Jobs?

There are some who might worry that open access to teaching materials threatens the livelihood of teachers. Some faculty members may wonder, "If people can watch videos of my lectures online or read my lecture notes, then why would the university keep me around at all?" This is a sensible concern, but as the history of libraries demonstrates, the vast majority of instructors have nothing to fear.

During the late 19th and early 20th century, Andrew Carnegie founded or endowed libraries all over the world—more than 2,500 of them. These libraries provided open access to collections of educational materials. Undoubtedly, most universities in the world today have libraries filled with collections of openly accessible educational materials. Why do faculty not feel threatened by university libraries? The logic should be exactly the same. Why do they not ask, "If students can read the lecture notes or textbooks of the very best minds in the world on each subject in the university library, then why does the university keep me around at all?"

The answer is, of course, that true teaching encompasses much more than the mere transmission of information. For most faculty, open educational resource initiatives are no more of a threat than the university library. Only instructors who provide no more interactivity to their students than would a book borrowed from the library need to fear greater access to information. Personally, I have always believed that any professor who could be replaced by a collection of educational materials deserves to be replaced by them.

The Future of Openness in Education

One of the key mechanisms underlying the success of the open source approach has always been at the core of higher

education's workings and values—peer review. Most academics are familiar and comfortable with blind peer review of their research. Nonblind review of the teaching practices and materials of instructors, as would be typical of open source protocol, is likely to be another matter. As uncomfortable a proposition as this new openness may be for some, I believe it is the future of higher education.

As I recently testified to the Secretary of Education's Commission on the Future of Higher Education:

Many in the public look up to Research I universities as the very pinnacle of higher education. It may be surprising, then, to hear that when faculty at MIT, USU, and other universities are invited to open their syllabi, lecture notes, assignments, and other materials for everyone to see, some faculty respond by asking first for time to tidy up their course materials. They are cautious because the move toward openness takes teaching directly into the heart of the scholarly world for the first time—it exposes teaching to the quality-increasing pressures of peer review. This openness also opens the materials to other kinds of review, creating an unprecedented level of transparency to all higher education stakeholders, including parents and alumni, with regard to an institution's teaching and learning activities. (Wiley 2006, 5)

Just as the principle of diversity and its attendant benefits have been the focus of a decade or more of concentrated advocacy within higher education, we must begin making a conscious effort to promote a culture of openness in all aspects of academic life—in our teaching, in the results of our research, and in the software and other tools we use to perform our work. Students, faculty, and staff around the world are awakening to the power of openness to transform higher education.

The world is changing in many ways as popularized by recent books such as *The World Is Flat* (Friedman 2005). Business, science, and other areas of society already leverage these changes to their benefit. In contrast, higher education has adapted too little in response to these changes and is consequently in very real danger of becoming irrelevant. Higher education's willingness and ability to evolve toward openness will be a strong predictor of its future relevance. If higher education is to fulfill its mission as set forth in visionary documents like the First Morrill Act (1862), which granted land to the states for colleges that would "promote the liberal and practical education of the industrial classes on the several pursuits and professions in life" (Section 4[8]), openness must become a core part of academic culture.

References

arXiv.org e-Print archive. n.d. Home page. http://arXiv.org/ (accessed September 30, 2006).

Athabasca University. 2006. Athabasca University moves to Moodle. http://www.athabascau.ca/media/index.php?id=132 (accessed September 30, 2006).

Eysenbach, G. 2006. Citation advantage of open access articles. *PLoS Biology* 4(5). http://rd.plos.org/pbio/0019.php?redir=0001 (accessed September 30, 2006).

Friedman, T. 2005. *The world is flat*. New York: Farrar, Straus, and Giroux.

Kuali Project. n.d. Home page. http://kualiproject.org/ (accessed September 30, 2006).

The Open University Media Relations Office. 2005. The Open University builds student online environment with Moodle and more. http://www3.open.ac.uk/events/7/20051124_40647_o1.doc (accessed September 30, 2006).

Public Library of Science (PLoS). n.d. About PLoS. http://www.plos.org/about/index.html (accessed September 30, 2006).

United States Congress. 1862. First Morrill Act. http://www.higher-ed.org/resources/morrill1.htm (accessed September

30, 2006).

Wiley, D. 2006. The current state of open educational resources. *iterating toward openness*. Weblog entry, February 3. http://opencontent.org/blog/archives/247 (accessed September 30, 2006).

Wiley, D. 2006. Testimony to the Secretary of Education's Commission on the Future of Higher Education. http://www.ed.gov/about/bdscomm/list/hiedfuture/3rd-meeting/wiley.pdf (accessed September 30, 2006).

Open Source/Open Course Learning: Lessons for Educators from Free and Open Source Software

by Robert Stephenson

Free/Open Source Software (FOSS) has transformed the software industry. As noted by other authors in this issue, academic information technology (IT) is already realizing many benefits by adopting open software; such benefits include reduced cost, absence of user restrictions and vendor lock-in, and consistency with traditional academic values of openness and sharing. The greatest benefit of the FOSS movement for educators, however, is not cheaper or better software but the model it provides of a social, cultural, and legal framework capable of harnessing IT to improve learning.

At this point, some may object: "Universities have been using IT for a half-century and hardly need a new model." But formal education has used IT principally to support administration and research and has been slow to adapt it to improve its core business of teaching and learning. Traditional learning is still too passive, too parochial, too hierarchical, and too artificial. By harnessing IT effectively, educators can make instruction more graphic, dynamic, and active than it is now. They can introduce students to real-world experts and real-world problems and create communities of practice that promote learning. Others may object that a huge amount of online content is already available at no charge, so open source learning is old news. But price is the least important issue in open source learning, as a review of the factors critical to the success of FOSS will make clear.

In the model I outline below, the characteristics of FOSS that have contributed to its rapid rise and success serve as the inspiration for grassroots, open-source learning communities—or more succinctly, *open course* communities—that would be capable of transforming education just as FOSS communities have transformed the software industry. To be sure, the participants and domains of FOSS communities are not the same as those within the open course communities I envision, and consequently the former provide a pattern for education to emulate rather than a precise blueprint. Nevertheless, in light of the similarities that do exist between these two worlds, I believe that the FOSS-inspired approach can revitalize educational practice.

How and Why Open Source Succeeds

Richard Stallman (1996), founder of the Free Software Movement, defined FOSS as follows:

"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer."

Free software . . . refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose . . .
- The freedom to study how the program works, and adapt it to your needs . . .
- The freedom to redistribute copies so you can help your neighbor . . .
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits . . . (¶ 2)

The license definition does not preclude a modest charge for the software, but the freedom to redistribute prevents monopoly rent and thereby limits the price that may be charged. With the near-zero cost of Internet distribution, the market price for downloaded material is often less than the overhead of a financial transaction and therefore effectively becomes zero.

The success of FOSS, however, is not directly due to any of the above freedoms but rather to the community that comes

along with FOSS's egalitarian philosophy. FOSS projects succeed—when they do—because they are embedded in a supportive community of practice, not just because they make their source code available (Weber 2004). FOSS communities organize and govern themselves, continually evaluate and improve their products, and grow in size and influence—generally in a bottom-up fashion with little or no external funding or institutional support.

Mozilla's Firefox browser, for example, owes its stunning success to its community: the teams of (largely) volunteer programmers and testers, the volunteers who write documentation or staff the help forums, and especially those who tell friends about it and help answer their questions. This support community plays a key role in driving adoption because by reducing the difficulty of mastering how to download and use the browser, it reduces adoption cost. Commercial software companies offer support, too, but it is usually less helpful (and often more expensive) because they cannot match the sheer manpower and enthusiasm behind many open source projects. Too often commercial support consists of a low-level employee reading from a script: "First, try rebooting the computer. Next, uninstall and reinstall the software . . ." The crucial difference is that successful FOSS communities grow spontaneously and become self-sustaining and self-supporting, recruiting new users who become members of the community themselves.

The Concept of Open Course Communities

Like their FOSS counterparts, open course communities are defined by their freedoms:

- The freedom to use material for teaching and self-study.
- The freedom to adapt material to meet students' needs.
- The freedom to redistribute material to help others.
- The freedom to release improvements so that the entire community benefits.

The freedom to adapt and improve is even more important for learning materials than for software. Adapting is more than just a matter of internationalization and localization. Learning occurs more readily in a culturally familiar context, so a Homer Simpson example that connects with community college students in Atlanta may be ineffective or inappropriate for medical students in Kuala Lumpur. Using learning assets in a new setting often requires that they be adapted. As Robby Robson puts it, "Context is the friend of learning and the enemy of reuse" (2003, "Context," ¶ 2).

Affording these freedoms is not sufficient, however, for open content to be embraced by most teachers. Fifteen years of institutional and philanthropic support has produced many digital libraries and large stores of free and open content, yet their impact on mainstream education remains slight. Why?

The answer is that zero price does not mean zero adoption cost. Adopting a new learning asset takes time and is in itself a learning process. Most teachers have more work than time and are wary of adopting new technology without knowing how long it will take to master. Their cost (expressed in terms of hours) is: Adoption Cost = Mastery Time (difficulty, experience) + Price per Instructor/Hourly Wage.

Mastery time is an increasing function of the difficulty of the materials and a decreasing function of the instructor's experience with them. The price of the materials can be nil, but mastery time is always greater than zero so it is usually the dominant factor in the cost of adoption. This observation is as true for software as for courseware, and FOSS's most important lesson is that a strong community can reduce this cost.

Open Course Communities: Key Ingredients of Success

Support and Collaboration

A strong support community is crucial in helping mainstream faculty members adopt online learning materials. The community can help teachers find materials that are right for their courses and students, and it can help show them how the materials might be used. It can evaluate the materials, creating a feedback loop for improvement. Community members can contribute back improvements, new learning assets, lesson plans, and assessment questions so that all

benefit.

Experience to date has shown that content alone—however excellent it may be—is not enough to impact formal learning significantly. Like the vacuum cleaner salesman's boast that "this machine will do most of your work by itself," it is false advertising. If educational content is to be effective for the mainstream, it needs to be supported by a strong community of teachers, teacher trainers, developers, technologists, and students. Learning is always the result of a conversation, however indirect that conversation may be. What is needed is not just open content but open course, defined as follows: Open Course = Open Content + Community.

Students benefit from this community not only indirectly but also from direct participation. Involving students in the evaluation and improvement of their class materials fosters more active and critical learning. Focusing on content alone perpetuates the idea that teaching is no more than feeding students safe, predigested bits of knowledge for their enlightenment. Although such an approach may be useful for building foundation knowledge, stopping learning at that stage creates a crippling dependency. More advanced learning is messy and open-ended, and in such contexts an open course approach is more effective pedagogically when it welcomes student participation.

An open course community is a knowledge ecosystem where teacher, courseware developer, librarian, and student roles are linked together by collaboration, sharing, and feedback. Like any ecosystem, it is characterized by circular flows of information, resources, feedback, and credit; the knowledge ecosystem develops its own, alternative economy based on exchange and reputation. Since the academy is already structured according to disciplinary boundaries, most open course communities are likely to be discipline-specific, but they will entail a much higher degree of collaboration—both horizontally across institutions and vertically between students and faculty—than is traditional in educational practice.

Assessment and Evaluation

Any successful complex system is the product of some sort of evolutionary improvement process. Engineering has well-developed standards of quality that software developers can use to choose between two competing programs. This is not the case in education where learning assessment is seldom standardized or tied to agreed-upon competencies and where different institutions and teachers are often fiercely loyal to their own ways of doing things. For open source learning to succeed, however, it still needs to address the issue of objective assessment. Just as many FOSS developers are moving to test-based development (Beck 2003), so should open course developers. This would require that

- each learning object have explicit learning objective(s);
- each objective have an associated battery of mastery questions, exercises, and problems;
- teachers administer these assessment materials to their students; and
- aggregated results be made available to the community.

This approach will enable the community to weed out ineffective assessment items and compare assets that share learning objectives based on the assessment results. This is, of course, no absolutely best way to teach a given subject because students are not all the same in any particular context; yet it is important to determine whether, for a particular group of students, this or that learning object is more or less effective in teaching a given subject. As Thomas Angelo points out, "assessment techniques are of little use unless and until local academic cultures value self-examination, reflection, and continuous improvement" (Angelo 1999, "Changing Our Mental Models: Assessment as Culture Transformation," ¶ 2).

A side benefit of this approach is the development of a large bank of vetted assessment materials classified by learning objective, a valuable open educational asset in its own right. For example, the question bank would not only allow for informed decisions about what learning object to use in a given situation but would also make it possible to compare the outcomes of teaching a topic with and without using learning objects. The evaluation I advocate is open, voluntary, decentralized, and bottom-up, and thus it avoids the controversy that has plagued standardized testing.

IT Infrastructure for Collaboration

These beneficial network effects I have described presuppose that many participants in an open source learning community contribute substantially to the community by developing learning assets, creating mastery questions, administering tests and submitting results, answering questions, helping faculty colleagues, submitting bugs and suggestions, and in a dozen other ways. Such contributions will occur only if they are easy and become a social norm.

Facilitating the human interactions that are important for learning and the growth of the community may require new technical tools. Open course communities need a technical infrastructure that makes collaborating with distant colleagues, finding the content needed for tomorrow's class, incorporating content into a lecture or online module, creating and administering student assessment, and returning anonymous data to the community easy. For students, the infrastructure needs to make using the online materials, taking assessments, interacting with the content developers, and generally contributing to the community easy. The role played by IT here is to enable collaboration, reduce friction, and facilitate data collection. It should also, like good plumbing, stay in the background and not become an issue.

Promoting Participation

Active collaboration is vital to the health and growth of the open course community, but there are obstacles to member participation. Most teachers are too busy with their workload, too uncomfortable with technology, and too unaccustomed to collaborating. To overcome these obstacles, the following elements are necessary:

- Barriers to participation and contribution need to be made as low as possible (through technology, careful planning, and thorough user testing).
- Active members of the community need to reach out to colleagues and encourage them to join the community.
- Educators need clear incentives to participate. These could be institutional rewards like release time or a stipend or increased status in the eyes of their peers. The most effective incentive, however, is a convincing demonstration that by actively participating, they will teach more effectively and save time in the long run.
- Educators need to understand clearly that their contribution is needed and in what ways they may give it. There should be enough ways to contribute to accommodate every talent and taste.

In addition, the license may include a giveback clause that makes helping an obligation (Exhibit 1). Specifically, for every hour that an educator's class spends using open course content, the teacher and students collectively owe an hour's contribution to the community. The open source learning communities are not in the business of policing compliance, however, and giveback is imposed only by persuasion and reputation.

Some will choose not to contribute while others will prefer to continue teaching in a more traditional way and eschew open course learning resources altogether. Those who do participate in the community, however, will contribute to the improvement of the open content and help recruit new colleagues. In the process, they will strengthen the community and all will benefit.

The Status and Future of Open Course Communities

Briefly, the main arguments made in this commentary are as follows:

- FOSS demonstrates that network effects occur at the edge of the network: in most cases bottom-up is more powerful than top-down.
- FOSS's lesson is that an active community of practice is the key to success. An open course collaboration is a knowledge ecosystem with an economy based primarily on exchange and reputation. When such a community involves all stakeholders, it not only provides the most value to its members but also grows the fastest.
- Including students in this community of practice strengthens their education.
- An open course community needs the ability to modify its resources since this is the only way they can

be improved or adapted for new contexts.
- Community resources will evolve only if they include assessment as an integral component and the results of this assessment are used to drive improvement.
- Stakeholders need lots of simple, easy ways to make helpful contributions to the community so that it becomes a social norm. Ways to promote this include incentives, a reputation system, and a license that requires contribution. Technology is needed to make these contributions as frictionless as possible.

No such community exists yet, although a few, such as the grassroots Harvey Project (Stephenson 2000), come close. Some top-down projects, such as the Global Education and Learning Community and the Social Authoring Project of the National Registry of Online Courses have similar goals.

Ad hoc solutions for most of the technical issues exist today. For example, MERLOT makes it possible to find and review large amounts of online content, and the eLearning XHTML Editor helps teachers create their own online courses using open content. OpenCourse.Org and Utah State University's EduCommons support collaboration around building, adapting, and evaluating open content. Commercial tools like Question*mark* and Respondus and the Sakai Project's open source tool SAMigo build and administer standards-based tests. What are still lacking are tools that are more integrated and more transparent for mainstream educators to use. These improved tools can be built by FOSS developers within the academic community. This is exactly the sort of "scratch your own itch" solution that FOSS has mastered in its approach to software development.

As these issues are solved, open course communities will begin to grow in size and improve in effectiveness. Eventually they will transform education, no matter how modest their beginnings.

[Editor's note: This article was adapted from a presentation at the Open Education Conference at Utah State University in Logan, UT, September 2005.]

References

Angelo, T. 1999. Doing assessment as if learning matters most. *AAHE Bulletin* 59 (9): 3-6. http://education.gsu.edu/ctl/outcomes/Doing%20Assessment%20As%20If%20Learning%20Matters%20Most.htm (accessed September 30, 2006).

Beck, K. 2003. *Test-driven development by example*. Boston: Addison-Wesley.

Robson, R. 2003. Reusable learning | design. http://www.reusablelearning.org/index.asp?id=30 (accessed September 30, 2006).

Stephenson, R. 2000. The Harvey Project: Open course development and rich content. In *Case studies on information technology in higher education*, ed. L. Petrides, 185-194. Hershey, PA: Idea Group Publishing Co.

Stallman, R. 1996. The free software definition. http://www.gnu.org/philosophy/free-sw.html (accessed September 30, 2006).

Weber, S. 2004. *The success of open source*. Cambridge, MA: Harvard University Press.

Getting Open Source Software into Schools: Strategies and Challenges

by Gary Hepburn and Jan Buley

This article focuses on the sociopolitical aspects of implementing open source software (OSS) in schools. Supporters of such implementation make a compelling case for OSS based on the clear advantages it offers over commercial applications—advantages such as its low cost, its freedom from cumbersome usage restrictions, and its ethical value in promoting a more democratic form of technological practice (e.g., BECTA 2005; Hart 2003; Hepburn 2004, 2005; Open Options 2005; Tong 2004). However, while these benefits make the transition from commercial software to OSS attractive, actually making the change is a complicated process. Below we will briefly describe some approaches that may be used when integrating software change; we will then indicate several sociopolitical factors that may act as barriers to such change and discuss how these barriers may be addressed in the decision-making process. Our intention is to provide suggestions for leadership and research to support schools as they consider moving to OSS-based computing environments.

Implementing Open Source Software

A wide variety of OSS is available for download, most for little or no cost. OSS applications exist for a wide range of operating systems including commercial platforms such as Windows and the MacOS as well as Linux, the nonproprietary flagship of OSS and a capable desktop alternative. Further, the comprehensive variety of functions afforded by OSS applications, which include everything from office suites to Web editors to graphic programs, is comparable to those afforded by commercial applications currently used in schools (Table 1). The rapid growth of OSS development has thus reached such a point that the continued use of commercial applications can no longer be taken for granted as the best option for educational institutions; open source systems now offer such institutions much of the same technological benefits without the added licensing cost.

Schools can harness the potential of OSS, provided they carefully plan for its implementation. This planning should account for the time-consuming technical process of changing a school's computing environment as well as the sociopolitical challenges that accompany such a change. Before exploring these challenges, we will outline three approaches for introducing open source software into schools.

1. *Open Source Applications on Windows.* OSS counterparts can be installed in place of commercial software and successfully run on the existing commercial operating system, such as Windows or MacOS. This approach significantly reduces software costs and releases schools from commercial software restrictive licensing agreements. The Linux operating system can be installed on a school's computers alongside Windows (or the MacOS). The hard drive of the computer is divided, allowing space for both operating systems and their applications. When a user boots the computer, she or he may choose the OSS or commercial operating system to use. This approach saves little or no money since it still retains the existing commerical software. Further, this approach provides little motivation for users to try the less familiar OSS option.
2. *All Open Source*. The most technically straightforward way to introduce OSS is to replace all commercial operating systems and software with Linux and open source applications. In this rapid conversion to OSS, the cost savings are dramatic.

When selecting an approach to implement OSS, schools must consider the impact of the approach on the school's computer users as well as on the technical and training staff. Despite the financial and ethical reasons for immediately implementing an all open source computing environment, this approach requires dramatic change in user habits and support requirements. To avoid the stress of an immediate all open source approach, a gradual approach may be considered. One successful strategy can be to use the open source applications on Windows approach—alone or in combination with the Linux and Windows approach—as a first step towards eventually adopting the all open source approach as an endpoint. This intermediate stage would provide users more time to adapt to the new OSS environment;

it would also provide technical and support staff more time to troubleshoot and configure new OSS packages and help users make the transition to OSS. This gradual process could make the final move to all open source relatively simple and stress-free.

What Is Stopping Us?

To identify the pace of change appropriate for a school, educational leaders should also consider the sociopolitical barriers to changing the computing environment. In our work promoting OSS and supporting schools interested in using it, we have observed several factors that can be obstacles to its implementation. These sociopolitical barriers indicate areas in which more work needs to be done by leaders to help schools plan their transitions to OSS; they also represent areas in which researchers can focus and perhaps gain a richer understanding of the dynamics of change in educational technology. These factors are by no means mutually exclusive—they are strongly interrelated and all are likely to be relevant to almost any instance of OSS implementation.

Awareness of OSS

One clear factor that continually arises in our discussions with inservice and preservice teachers, administrators, teacher educators, and sometimes even technology leaders is that few in the educational community are aware of what OSS is and the potential it holds for education. In fact, most educators have not heard of OSS and most of those who have only possess a vague notion of what it is. The problem with this lack of awareness is obvious: There is little possibility of significant adoption of OSS without a greater understanding of the technology. Educational leaders must become aware of the OSS option in order to initiate and facilitate any movement toward making this option available to teachers and students. Even when leaders do become more knowledgeable about the OSS option, teachers must also become familiar with the reasons for using OSS and the vast array of high quality software available so that they can be enthusiastic and confident supporters of its use in the classroom.

In this context, the role of educational researchers and scholars remains indispensable for promoting greater awareness of OSS and establishing practical precedents for its widespread adoption. A number of cases of successful OSS implementation and its associated benefits have been documented in professional newsletters, the popular press, and elsewhere (e.g., Canada's SchoolNet 2003; Cutter Project n.d.; Gedda 2006; Leete 2006). In turn, the recent report from Becta (2005) has taken a significant step forward by bringing together several OSS case studies and adopting a more rigorous research methodology to assess their results. Reports like these have the potential to convince educational leaders and others in the education system that there is a clear rationale for OSS, that there is a growing knowledge base of actual implementation efforts, and that OSS consequently needs to be considered as a viable option for institutions. The problem is that this vital information is often not reaching educators. Governments and educational researchers have to take up the role of collecting, analyzing, and disseminating the data so those who need to learn more about OSS can do so.

The role of researchers in turn may be understood as part of a three-tiered effort for improving OSS awareness in the educational community. Such research can make the case for OSS by identifying strategies for successful implementation. Educational leaders can then use this research in combination with preservice education to develop training programs for teachers. Finally, through inservice and preservice education, teachers can become familiar with OSS in general as well as specific OSS applications that are the most useful for teaching and learning activities.

The Politics of Payment

Perhaps the most compelling argument for implementing OSS is its potential to reduce a school's software costs drastically. This argument, however, can sometimes be less persuasive in schools where computing costs are distributed over three levels of school administration. In the schools we work with most directly, software costs are covered at the provincial level by the Department of Education, at the district level by the school board, and on the local level by the schools themselves. For example, the province may purchase site licenses that allow all the schools in the province to use particular commercial software packages; in turn, the school boards often choose to license other software packages for use in their schools; finally, individual schools may choose to license any additional software that they would like to

have even though the province or board did not purchase these licenses. The net effect of this situation is that people working at any one of the three levels are not directly impacted and are often unaware of the full cost of purchasing software licenses. Although there is a great deal of money spent on software within the system, no one group feels the full burden of the expense.

To address the issue software costs carefully and thoroughly, all levels must understand the overall impact of expensive software and the benefit of reducing those costs. Studies like the one carried out by Becta (2005) promote such an awareness by taking into account the full costs of software licenses at the schools studied as well as other technology-related costs; in turn, such studies should be expanded in scope so that the cost savings of OSS can be assessed further at the systemic level as well as the local level of the individual institution. With appropriate dissemination, research in this vein can provide educators with all the rationale they need to begin exploring OSS. The dissemination of such research needs to be carried out broadly with a special emphasis on government leaders who are responsible for ensuring that public money is well spent. These leaders can ensure that research and reports find their way into the hands of those who need to see them. The cost savings that result from using more OSS within the system can impact more than just technology; this money can be directed toward other necessary programs and resources to enhance learning. Demonstrating such associated benefits in cost-benefit studies of OSS can also go a long way towards convincing key decision-makers to take steps towards implementation.

The Dynamics of Change in Schools

Donald Murray (1989) argues that what is certain about teaching and schools is change; educational technology is no different. The complexity of these changes requires that they be approached on a system level. Unfortunately, the complicated environment between the various levels and the groups involved make systematic change difficult no matter how well justified it may be. Those of us who would like to see more OSS in schools need to appreciate this and be patient. It is helpful to think of OSS implementation as a gradual process rather than one that happens quickly.

Change at any level requires some degree of vulnerability, and supporters of OSS in schools must be prepared to address the vulnerability of groups at every level of the educational system. For example, school principals are sometimes worried about how their hardworking teachers will adapt to new software. Within the school system, a great deal of effort has been devoted to getting teachers ready to use educational technology in the classroom. Many principals are reluctant to make additional changes to the software since this may discourage teachers, regardless of whether those changes make sense in the big picture. Furthermore, principals sometimes worry about whether school boards would be willing or able to provide technical and training support if such changes are to be made. Concomitantly, those at the board or government levels may be hesitant to integrate changes that may not be readily adopted in schools. In our research we have discovered that teachers and administrators are often reluctant to be put in positions where they themselves are the questioners. Too often, particularly in the areas of technology, educators are afraid of the unknown and do not want to appear foolish in front of their peers and students (Lapp et al. 1998). Making changes to a school's software stirs up such fears and we need to be cognizant of this. At the same time, we need to support risk taking and acknowledge the time investment required for educators to become familiar with a new software environment.

Insuring that groups at every level of the educational system understand why change is needed is a necessary first step. Once that is done, it is crucial to ensure that all groups are included in the process of planning, undertaking, and evaluating OSS implementation—through technology committees, for example—to ensure that their respective concerns continue to be addressed and that they remain fully invested in the process. Furthermore, instructors who have to bear the burden of such change must be encouraged and supported as they do so by such measures as the allotment of training time, expanded access to support staff, and formal recognition for their accomplishments. While some measures of faculty support could require expenditures that initially compromise the cost savings of OSS in the short term, such measures should be regarded as investments that will eventually be offset by the long-term savings afforded by the new technology. Finally, researchers will need to be involved in studying these strategies in order to determine their relative value in making the implementation process a smooth and successful one for all involved.

Understanding Software Choice as an Ethical Consideration

Software is often seen as ethically neutral, a resource that is simply present rather than one that is the result of choices—choices that impact people within and beyond schools. However, many have argued for the adoption of OSS in schools on ethical grounds (Hart 2003; Hepburn 2004, 2005; Open Options 2005). Claims that using OSS would strengthen the ethical positioning of schools emphasize OSS's role in promoting digital equity, eliminating commercial messages from school environments, and promoting the development of an educational commons. Because OSS is free or low cost and because open source licenses ensure that anyone has the right to use, redistribute, and modify the software freely (OSI n.d.), the way in which such software may be used makes it more democratic than commercial software. Scholarly leadership can be helpful in developing and promoting the idea that schools have an obligation to promote an egalitarian learning environment and in encouraging educational leaders to apply a critical perspective towards software to the same extent that they do towards other educational resources.

Technical Personnel

Most school systems currently have a reasonable level of technical support. The personnel responsible for providing this support are crucial to keeping the software working well and ensuring that software changes go smoothly. Given that few educators and administrators are familiar with the technical details related to changing a school's software environment, the opinions of the technical support staff may have a great influence on decisions related to implementing OSS. However, because schools have been heavily dependent on commercial software, the support personnel may have limited experience with OSS. As a result of this inexperience, support staff may be somewhat reluctant to suggest or endorse a plan to use OSS as it may require considerable retraining. When considering the use of OSS software in schools, it is important to find technical support staff who are not only knowledgeable of the school's current hardware and software systems but who are also experienced in OSS and fully prepared to work with it. This may require continued training and/or making experience with OSS a qualification for new hires.

Conclusion

Those of us familiar with OSS can see the clear benefits it holds for education. The dramatically lower cost of the software along with greater freedom in how it is used and how it is shared make it a valuable resource for schools. With OSS and its related know-how readily available, acquiring and installing OSS is the easy part. The real challenge is in helping educators, administrators, and technical support personnel realize the huge potential OSS holds for teaching and learning and in planning an appropriate implementation approach that addresses the sociopolitical barriers we discussed earlier. Future research on how to plan and implement OSS in schools effectively will not only heighten awareness of OSS, but will hasten its adoption.

References

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. http://www.becta.org.uk/corporate/publications/documents/BEC5606_Full_report18.pdf (accessed August 26, 2006).

Canada's SchoolNet. 2003. Considering changing your school's computer system to open source software? One school's conversion to Linux has been a complete success. http://www.schoolnet.ca/today/article_2003-06.asp (accessed August 26, 2006).

Cutter Project. n.d. Orwell High School case study. http://www.cutterproject.co.uk/Casestudies/orwell_high_school_cutter_case_study.php (accessed August 26, 2006).

Gedda, R. 2006. Sydney school teaches with Linux monopoly. *Computerworld*, March 21. http://www.computerworld.com.au/index.php/id;407925021;fp;2;fpid;1 (accessed August 26, 2006).

Hart, T. D. 2003. Open source in education. http://portfolio.umaine.edu/~hartt/OS in Education.pdf (accessed August 26, 2006).

Hepburn G. 2004. Seeking an educational commons: The promise of open source development models. *First Monday* 9 (8). http://www.firstmonday.dk/ISSUES/issue9_8/hepburn/index.html (accessed September 30, 2006).

Hepburn, G. 2005. Open source software and schools: New opportunities and directions. *Canadian Journal of Learning and Technology* 31 (1). http://www.cjlt.ca/content/vol31.1/hepburn.html (accessed September 30, 2006).

Lapp, D., J. Flood, and D. B. Martin. 1998. Teachers online: Using personal visual literacy skills to enhance professional teaching knowledge. *The Reading Teacher* 51:702-705.

Leete, G. 2006. Switching art students to GNU/Linux. *NewsForge*, March 22. http://business.newsforge.com/article.pl?sid=06/03/09/2238246 (accessed September 30, 2006).

Murray, D. 1989. *Expecting the unexpected: Teaching myself—and others—to read and write*. Portsmouth, NH: Heinemann.

Newman, J. 1991. *Interwoven conversations: Learning and teaching through critical reflection*. Portsmouth, NH: Heinemann.

Open Options. 2005. Making decisions about open source software for K-12. http://www.netc.org/openoptions/ (accessed on September 30, 2006).

Open Source Initiative. n.d. Open Source Initiative. http://www.opensource.org/ (accessed September 30, 2006).

Tong, T. W. 2004. Free/open source software: Education. New Delhi, India: Elsevier. http://www.iosn.net/education/foss-education-primer/fossPrimer-Education.pdf (accessed on September 30, 2006).

Vision 2010:
The Future of Higher Education Business and Learning Applications

by Patrick Carey and Bernard Gleason

The global software industry is in the midst of a major evolutionary shift—one based on open computing—and this trend, like many transformative trends in technology, is being led by the instructional technology (IT) staffs and academic computing faculty of the higher education industry. The elements of this open computing approach are open source, open standards, open architecture, and open communities. Such an approach does not only involve the use of open source software code; the application code (whether proprietary or open source) must also be developed to adhere to open data and content standards, designed to operate within an open architecture, and, in most cases, designed and developed based on requirements determined by a larger open community. Together, these elements hold the promise of the next generation of business and learning applications for higher education. If history does indeed repeat itself, we believe that this trend will have the same effect on most, if not all, other industries.

Such a trend may be seen not only in the emergence of various open software applications over the past few years but also in the adjustments that proprietary developers have made in their own business models. Future-focused commercial vendors are embracing open standards and open architecture, even when they are not open-sourcing their application code. In doing so, they recognize that open standards and open architecture simplify the processes of developing or acquiring new applications—and, in turn, of integrating and customizing client systems to match evolving business practices. They see that basing applications on open standards can accelerate the deployment process, ease adoption by more users, improve operational productivity, and reduce costs. They are also responding to significant patterns in user behavior that have already begun to change the playing field for software developers. For example, in analyzing the results of its February 2006 survey of U.S. higher education chief information officers (CIOs), the Alliance for Higher Education Competitiveness (A-HEC) found that "two-thirds of CIOs have considered or are actively considering open source with a leading edge of about 25% of all institutions actively engaged in implementing higher education specific open source applications of some type"; in response to these trends, the A-HEC concluded that the "value proposition for open source applications can be summarized as a combination of cost, control, and possibility for innovation" (Abel 2006, 5). In the commercial world, InformationWeek's November 2004 survey of more than 400 IT professionals found that although only 2% across all industries were using open source software primarily, two thirds were actively using open source, and three in five reported an environment of both open source and commercial software (D'Antoni 2004).

Such findings are consistent with our own observations at IBM. As a major and long-time participant in the open movement and as one of the world's largest software companies, we expect the future technological environment to be made up of a combination of open and closed source code, all of it adhering to open standards, thereby reducing the costs of integrating and operating the software by client institutions. At present, however, many institutions that have successfully implemented enterprise resource planning (ERP) systems and course management systems (CMS) still face the challenges of streamlining their operating procedures, reducing the financial and operational burdens of such systems, and ensuring that such systems can be made flexible enough to adapt to new business needs and technological innovations. Colleges, universities, and commercial software providers alike will need to formulate a clear vision of the factors that for the next five years will shape business and learning applications as well as the architecture in which the applications operate. To this end, we outline and discuss several key developments on the horizon of open computing that university planners should anticipate as they envision the future of their technological infrastructure.

The Future of the Open Movement in Education

In the next five years, what business models will higher education institutions adopt? What direction will application software development take?

Many predictable outcomes and trends will emerge. For colleges and universities to stay competitive, offer exceptional learning experiences, attract professors and students, promote research, and compete globally, they should consider adopting an open approach in designing or obtaining both business and learning application software. An open approach has at its core a focus on reducing the costs of the integration and operation of campus business and learning systems,

empowering the institution to redirect scarce resources into improving the academic and research environment. The evolution of an open-based movement in higher education will unlock the innovative talent, forces, and flexibility that institutions desperately desire. The following trends will be particularly important for university planners to recognize as they formulate strategies for the future.

Modular not Monolithic

Applications developed an open, standards-based architecture are often assembled from components that are either freely available or less costly than comparable proprietary packages. Instead of relying on monolithic design as in the past, developers using an open approach rely on interorganizational or intraorganizational networks on the Internet to find functional modules loosely coupled as sets of services within a service-oriented architecture (SOA). The architecture is deemed service-oriented because applications and functions within applications can be joined and choreographed to create composite applications and business processes; this approach allows components to be selected and joined to create more precisely the composite applications and processes needed by an institution.

Reusable component software is available either at a price or for nothing from a variety of sources such as commercial independent software vendors (ISVs), online open source repositories, or subscription from a hosted service. For example, higher education groups such as the Java Architectures Special Interest Group (JA-SIG) sponsor open source projects and host repositories of applications and documentation that are freely available for downloading by all interested colleges and universities. In turn, IBM developed a framework for Java development called Eclipse and, rather than compete with other Java tool providers and require each company to create its own framework, contributed the code as open source and funded the start-up of the independent Eclipse Foundation. Today all major Java toolmakers are members of the foundation; each company is saving money and is able to rely on access to a consistent environment for open source development.

Through such open approaches, ISVs still create, assemble, sell, and support software solutions, but vendor differentiation shifts to innovation in applications and quality of support, particularly in integration services. IBM and other major vendors of enterprise resource planning solutions are moving in this direction, and next-generation course management systems such as the Sakai Project are already emerging to meet similar needs. However, continued progress is likely to be constrained by the requirements of the installed base and by the commercial vendors' desire to protect their investment in existing customer relationships and license maintenance streams. Consequently, university planners will need to exercise discretion in the systems they adopt; if a compatible open source solution cannot be found to address a particular need, they should still select proprietary software that utilizes open standards within an open architecture and thereby allows for modular development.

Community Open Source Model

Today, groups of colleges and universities are pooling application development and financial resources to create common sets of applications. This approach is called community open source. The community open source model is built on the basic principles of open source software development—that is, code is freely distributed and users are free to modify it. The main distinctions of the community source model are the methods for funding, managing development, and guaranteeing sustainability.

The community source model is favored by the Andrew W. Mellon Foundation and is exemplified by the Sakai Project (course management system) and the Kuali Project (financial system). Both projects leverage code already developed by universities to jumpstart the creation of a new breed of open source enterprise applications. Their model functions in the following manner:

- Foundations provide start-up funding to be matched by participating institutions.
- A core group of institutions supplies oversight, central development resources, and funding.
- A larger group of institutions makes smaller annual financial contributions and participates in defining requirements, testing, and evaluation.
- Core developers and project leaders are compensated for their work.

- Commercial partners contribute technical resources and offer customer support services.

Institutions seeking to benefit from the possibilities of open source computing should give special attention to projects such as these since they have established a strong foundation for sustained innovation and lasting value. These value networks, sometimes unforeseen and unpredictable, will continue to sprout up within the higher education community as individuals and units with common interests and requirements come together. Consortia and associations of institutions (e.g., all colleges in a conference) that already have common-value structures are likely to be the early winners in sharing resources and exercising group influence and buying power.

Meanwhile, for all community open source projects that adhere to the concepts of an open approach—open standards, open source, and open architecture—the next logical step is the creation of a coordinating body, similar in concept to the Apache Software Foundation. Ira Fuch, Vice President for Research in Information Technology at the Andrew W. Mellon Foundation, similarly advocates the formation of an open source congress to coordinate the interproject collaboration, communication, and certification of multiple related projects within the higher education ecosystem (cf. Currie 2005). Such a coordinating body helps ensure the placement of a system to maintain and enhance the code over time, a major concern to any campus administrator considering an investment in a critical business system. As further advances are made to establish sustained coordination and development of open source projects, institutional planners will be able to make informed decisions about the potential adoption of such technologies.

Open Integration Framework for Education

One of the biggest challenges facing application developers is likely to be integration and interoperability—getting applications to talk to each other. Institutions can purchase or internally develop an SOA, but colleges and universities are not going to be able to quickly re-engineer existing applications and processes to be SOA-compliant and compatible with one another. The answer to this challenge lies in the creation of a new breed of open applications that will provide a flexible and scalable technical framework for supporting open standards and open application interfaces.

At the heart of an SOA is something called an enterprise services bus (ESB), a kind of middleware—software that operates between an operating system and an application—that utilizes standard data definitions and messaging formats to transfer data securely between different programs. An ESB supports the running of existing applications while adding new functionality and adapting to a new architecture based on Web services, and it is likely to be the first step in the shift to the full adoption of SOA and a modular design for connecting applications. The ESB will allow clients to follow an evolutionary approach in moving from a dependence on a single software provider or a proprietary technology platform to an open computing approach.

We strongly believe that an open integration framework for higher education will emerge that is based on an open source ESB from IBM and perhaps others. The open integration framework will employ commodity middleware, will support an extended set of trusted services, and will be customized to support the special data and content standards for higher education business and learning. A logical first instance will be in the area of student services—a Student Services Bus. The Student Services Bus, in combination with an existing student information system and learning management system, will provide a platform for integrating new services with core functions; in doing so, it will free institutions to think and act beyond the boundaries of traditional student learning systems and beyond the notion that all functionality must be delivered by one provider (cf. Carey and Gleason 2005). As university planners seek ways to improve their technological infrastructure, the flexibility afforded by advances in ESB technology will open new possibilities for easing the transition to an open integration framework and enhancing the services they provide their students.

Superior Vertical Solutions

Until now, the commercial vendors of enterprise planning and learning software have promoted a common code base across all industries as a way to minimize their code development and maintenance costs. This rigid one-size-fits-all approach often restricts functionality, causing unnecessarily high costs and forcing institutions to adapt their business

processes to the vendor's solution, instead of the other way around. As a result, educational institutions are often forced to change their existing business processes significantly or undertake expensive software customization projects.

However, in the new, open, and component-driven SOA, developers can begin to create or obtain superior solutions as customizable modules, which are designed as services to be plugged into the open, standards-based application architecture. The modular design makes it easier for developers to innovate and to create superior components without worrying about the total application system. For example, if one institution creates a Web-based course registration module that is superior in functionality—a module perhaps equivalent to Amazon in completeness and user-friendliness—another institution could adopt it easily by swapping out just the course registration module without having to replace the rest of the institution's existing student information system. If another institution creates an online testing module for a learning management system that requires minimal user knowledge of specialized design code, another institution could adopt it within its own learning management system without the need for replacing or extensively modifying the system itself. In such cases open architecture and open standards provide the foundation for greater flexibility as designers can incorporate different components from one system to the next.

Software-as-a-Service

Campus constituents—students, faculty and staff—are increasingly dependent on information and learning services provided by the colleges and universities. Their experiences with information and communications technology in other aspects of life have led them to expect reliability and 24/7 availability in campus systems. These requirements are pushing higher education institutions to provide additional support, which entails an increase in associated costs with no new supporting revenue. In some cases institutions have tried to reduce costs by deploying no-frills versions of ERP and CMS applications that do not require the same level of support. Yet by settling for lesser capabilities, such institutions may be compelled to sacrifice quality in order to remain relatively viable in the highly competitive area of student services.

In light of these challenges, many colleges and universities will need to rely on third-party providers in order to meet student demands more economically and on a level that cannot be duplicated by an individual institution. As they do so, an open architecture that utilizes open and universal security standards will serve as a crucial foundation to facilitate such partnerships.

1.) *Applications*—As colleges and universities look for new ways of controlling costs, reducing the burden on their technical staff, and simultaneously improving constituent services, we can expect resurgence in hosted ERP and CMS applications. More importantly, the existence of an open architecture will facilitate the outsourcing of portions of the application base, and institutions will easily be able to swap out third-party service providers if they are not satisfied with the quality or price of the service. Likewise, such an open architecture would allow administrators and instructors to select from a variety of competing third-party providers in the design, customization, or support of specific CMS applications.

2.) *Integration as a service*—A major concern in adopting a service-oriented architecture is the effort required to maintain, manage, and monitor many application connections. Rather than worrying about services with a multitude of interfaces, higher education institutions may begin to consider outsourcing the integration layer. All connections to business processes from the institution would conform to a set of defined higher education industry standards for data, messaging, privacy, and security and would connect to an integration provider. Integration providers would, in turn, manage the service-level agreements and transactions with partner networks.

3.) *Outsourced processing*—Generally, every college and university performs the same administrative functions and has similar processes to support those functions. For example, every undergraduate admission process collects the same types of information, electronically and manually. Instead of collecting and scanning forms and translating data files in the back office, this work can be outsourced easily and cheaply to an agency that performs the same tasks for multiple institutions. In such cases, of course, institutions would ensure student privacy through public agreements with the service provider, and with such agreements in place, office professionals would find much more time for personally serving their constituents.

Conclusion

An open computing approach will generate practical innovation in the business and learning areas of higher education institutions along with the cost savings that colleges and universities are desperately seeking. Colleges and universities that aspire to be regarded as one of the best places to work or be educated must look beyond today's applications and operating models; these higher education institutions must be committed to the continuous evaluation and adoption of new technology innovations and to the pursuit of opportunities for collaboration in what has been historically a very collaborative environment. Developing a long-term strategy for administrative and learning applications is one of the most important financial decisions campus executives can make as it will have a significant impact on all members of the community for decades to come.

References

Abel, R. 2006. Best practices in open source in higher education study: The state of open source software. Lake Mary, FL: The Alliance for Higher Education Competitiveness, Inc. http://www.a-hec.org/media/files/A-HEC%20open%20source%20hed%20030106.pdf (accessed September 17, 2006).

Carey, P., and B. W. Gleason. 2005. Student services system—Next generation. White Plains, New York: IBM Corporation. http://www-03.ibm.com/industries/education/doc/content/bin/IBM_BCS_White_Paper_Student_Services_System_FINAL.pdf (accessed September 17, 2006).

Currie, C. 2005. The open source congress. *EDUCAUSE Review* (July/August): 80-81. http://www.educause.edu/ir/library/pdf/erm05414.pdf#search=%22fuchs%20%2Bopen%20source%20organization%22

D'Antoni, H. 2004. Open source software use joins the mix. *InformationWeek*, November 1. http://www.informationweek.com/story/showArticle.jhtml?articleID=51201599&tid=5979 (accessed September 17, 2006).

Harnessing Open Technologies to Promote Open Educational Knowledge Sharing

by Toru Iiyoshi, Cheryl Richardson, and Owen McGrath

The Knowledge Exchange Exhibit and Presentation (KEEP) Toolkit, a set of software tools designed to help educators provide focused, detailed investigations and demonstrations of effective teaching practice, was developed in 2001 by the Knowledge Media Laboratory (KML) of the Carnegie Foundation for the Advancement of Teaching as a specialized in-house software resource. However, within five years this software steadily evolved into the hub of a distributed community of over 10,000 users, and in 2006 it was released as open source—thereby making it possible for individuals and institutions anywhere to participate in this community. Encouraged by the rapid usage growth but concerned about sustainability, we saw that the best bet for long-term viability in the software would come from inviting our user community to participate in the KEEP Toolkit's future development.

In what follows we first provide an introduction to the key features of the KEEP Toolkit, illustrate its early application as a means of promoting shared inquiry into pedagogical practice, and address its role as a tool for documenting the pedagogical value of learning objects. We then discuss the factors that led us to pursue an open source approach to further software development, and we describe the stages that characterized our implementation of this approach. While potential users may find primary interest in our account of the software itself, we believe that developers, designers, and planners may find our account of the open source transition helpful as a roadmap for their own potential efforts in this direction.

The KEEP Toolkit: Development and Functionality

Over the last several years, the KML has been devising ways to take advantage of emerging technologies and new media to transform what teachers know and do. In 1998, the KML started working in collaboration with partners at the Carnegie Foundation to design Web-based portfolios that demonstrate how teaching practice and student learning can be documented with multimedia and then shared on the Internet. Inspired by the increasing interest of faculty, programs, and institutions to develop and use these portfolios for collective knowledge building, the KML subsequently set about creating a toolkit that instructors and students could use in order to make visible the experiences of teaching and learning that permeate instructional settings every day. The resulting KEEP Toolkit has become an economical and accessible means of achieving this goal, making it possible for users to take advantage of Web technology in order to share their work and reflections on their work.

The primary functions of the KEEP Toolkit provide educators with the ability to create Snapshots, or succinct online overviews of teaching and learning experiences, along with reflections, supplements, and related resources. Creating Snapshots involves working through a set of Web-based forms that allow users to upload artifacts and evidence of teaching and learning (e.g., student work, their own reflections, sound files, pictures, videos), use either pre-established or personally-created templates that quickly organize those materials, and share the Snapshots with others in visually appealing and intellectually engaging formats. (Click here for an interactive demo illustrating the software.) Snapshots are delivered primarily online as Web sites, but they also can be distributed in printed form as handouts and posters. The underlying design philosophy of the KEEP Toolkit is that creating engaging Web representations of teaching and learning and sharing them effectively may always be intellectually challenging but need not be technically challenging.

Using the KEEP Toolkit to Share Inquiry: The CASTL Scholars Program

The first group of educators to utilize the KEEP Toolkit included participants in the Carnegie Academy for the Scholarship of Teaching and Learning (CASTL). Initiated the same year that the KML first began offering its Web-based tools to the Carnegie Foundation, the CASTL program was established "to support the development of a scholarship of teaching and learning that: 1) fosters significant, long-lasting learning for all students; 2) enhances the practice and profession of teaching; 3) brings to faculty members' work as teachers the recognition and rewards afforded to other forms of scholarly work" (2006, ¶ 2). To support these goals, KML first developed the Snapshot Tool in 2002; since then, the tool has been further modified as part of the KEEP Toolkit, which has become a key means for the program to fulfill its mission.

The value of the KEEP Toolkit in this context derives largely from the overall emphasis on scholarly collaboration in the CASTL program. In the program, faculty participants from various universities and colleges convene at the Carnegie Foundation to explore and share pedagogical questions, processes, strategies, and outcomes in face-to-face meetings that occur in six-month intervals over the course of 12-18 months. During these meetings the participants provide each other with support and feedback on knowledge of teaching that has been exchanged online, and the CASTL leadership team helps the participants frame effective questions about their teaching practice and walk them through the process of sharing it with others. The KEEP Toolkit serves as the technological foundation for such shared inquiry throughout the duration of the program.

The specific applications of the KEEP Toolkit in this setting may illustrate its usefulness as a structural enhancement to scholarly investigation as well as a means to support highly detailed forms of information sharing. The role of the software as a structural enhancement to investigation arises from the provision of specialized templates that allow program participants to refine the focus of their work as they use the Toolkit in the program. Before the first meeting, for example, the leadership team sends instructions on using the KEEP Toolkit along with a preliminary template that directs scholars to introduce themselves and their work. Between each of the subsequent meetings, scholars are given four more templates that provide benchmarks for their ongoing inquiries into teaching. These latter templates carefully prompt scholars to share specific information with sets of questions that are similar to those that inform a research proposal ("What is your issue?"; "How will you investigate this issue?"; "What resources do you need?"; "What is your evidence?"). Scholars answer these questions in the template itself by typing their responses in designated boxes. This process of responding in a Web-publishing format guides academic professionals in thinking about teaching and learning in scholarly ways and reflecting on teaching candidly.

Once program participants begin to define and pursue their respective areas of investigation, the KEEP Toolkit also offers them a valuable means of incorporating a range of media in their work; in this respect, the flexibility of the software is as vital as its accommodation of predesigned templates. Instead of struggling to produce a lengthy and linear report of an educational experience, program participants construct presentations that are enhanced by multiple forms of media such as interviews, images, and videos that provide engaging, detailed illustrations of pedagogical methodology and practice. The inclusion of these kinds of artifacts thus helps scholars move beyond the anecdotal as they indicate and present observable classroom exchanges for discussion. The KEEP Toolkit likewise encourages scholars to be succinct in their explanations by allowing them to use linked information and uploaded files to provide multiple layers of support for their conjectures. Rather than describing certain practices or processes, they have much more freedom to illustrate such practices and processes directly in their presentations. For example, CASTL scholar Whitney Schlegel employed the software to create a gallery of four semesters of student work; with this presentation she explored the potential value of "seeing" student learning of physiology through their layered, team-built snapshots. Other KEEP Toolkit case studies can be found at the Gallery of Teaching and Learning Web site.

In turn, the online availability of such materials both sustains and expands the process of shared inquiry in the CASTL program. The interim reports and presentations are shared with other program participants in a Sakai-based workspace (cf. Downes 2006); this workspace is an online community environment that serves as one of the primary meeting places, resource repositories, and information portals for all cohorts of CASTL scholars. The workspace is used for communication, collaboration, and documentation of the work completed during the course of the program, allowing for tentative ideas, successes, and challenges to be addressed at every step of the process. Furthermore, by taking advantage of the KEEP Toolkit as well as other features of the workspace—a wiki, a discussion forum, chat rooms, and other tools—CASTL scholars can sustain their investigations and explorations long after the program has ended. Every few weeks new resources and Snapshots are added to the workspace, and fellow scholars comment on them. At the end of the fellowship, many scholars choose to make their inquiry process public to their universities and peers by linking to Snapshots they created during the CASTL Fellowship. In this way, knowledge about particular teaching dilemmas is shared beyond the circle of faculty who initially discussed them.

The use of the KEEP Toolkit and Sakai-based workspace to document and share teaching inquiries thus enables a select group of faculty members to explore, improve, and build teaching knowledge. They engage in a rigorous process of learning to inquire about teaching, they discuss with others their dilemmas and processes, and they use these tools to document and make these processes visible and expandable.

Using the KEEP Toolkit to Share Practice: The MERLOT Template

In light of the usefulness of the KEEP Toolkit for participants in the CASTL program, the KML and its partners at the Carnegie Foundation soon began to consider how this resource could be used to supplement teaching materials created for online sharing and thereby provide pedagogical knowledge regarding the use of such materials. One key step in this direction came from their decision to establish a partnership with MERLOT; through this partnership, contributors to the MERLOT repository are encouraged to accompany their learning objects with KEEP snapshots that provide clear documentation of the pedagogical thinking that informs such objects. By providing such additional information, any contributor can ensure that others have a greater sense of the opportunities, challenges, and successes they may face in the use of a given learning object.

The specialized MERLOT template provides a key foundation for such contributions to the repository. When contributors submit a learning object to the repository, they can use the KEEP Toolkit to describe their motivation for creating the object, the learning activities associated with the object, the impact of the object on student learning, and their reflections on its use in the classroom. As they do so, the MERLOT template provides them with specific prompts that require succinct and direct explanations of how the object serves to enhance teaching practice. Moreover, as they fill out the template, contributors often insert links to definitions, assignments, resources, and other materials to enhance their brief descriptions and provide further information for a potential user of the object. All of this information helps a user understand the kinds of learning issues that the experience is meant to address, the type of environment that will ideal for learning, and the kinds of outcomes someone might expect. For example, one MERLOT contributor used the template to indicate how a particular music simulation could be used to support instruction not only in music but also in other disciplines such as biology and literature.

This partnership with MERLOT has been beneficial for faculty contributors as well as for other faculty who may be considering the use of such objects. The process of writing about the context of using an object has helped authors step back and think about their teaching and not just the content of their courses. The process has also prompted faculty to think about the relationships among content, teaching practice, and student learning beyond the ways of their specific academic disciplines; faculty then share this information so that others can learn from and effectively use the learning objects they find in MERLOT. Although learning object documentation via KEEP Toolkit has not been fully integrated with the repository itself, we anticipate that such documentation will become a standard feature of MERLOT.

Developing an Open KEEP Toolkit

These early experiences with CASTL and MERLOT reveal that besides making it possible for teachers to share aspects of their teaching in a single Snapshot, the KEEP Toolkit activities also serve as an opportunity for them to reflect on their teaching over time across sequences and collections of many Snapshots. Understandably, we sought to continue developing the tools to support such activities and bring them to a larger group of users. However, the Toolkit hosted on KML's server became popular so quickly that our original server design for storing and managing its content proved inadequate. As the number of contributions grew to many thousands, sharing and viewing content became slow and difficult.

Meanwhile, as the KEEP Toolkit experienced these growing pains, the open source trend was also on our minds. Efforts such as the Sakai Project and the Open Source Portfolio (OSP) Initiative were gaining attention in higher education as institutions began to consider shifting away from commercial or homegrown software systems (Wheeler 2004). It was at this time that we chose to make our own move toward "openness"—open source and open content—for the KEEP Toolkit. We made this decision not out of a desire to be trendy. The move to open source, we believed, was a necessary development that would allow other institutions to contribute resources (e.g., developers, designers, project managers) towards further improving the KEEP Toolkit. In addition, the move toward openness better suited the purposes and the general philosophy of KEEP. The sharing of code, for example, parallels the intent of sharing teaching resources. Allowing for collaborative development of new features should result in innovative approaches to creating and searching new KEEP content. Likewise, and practically speaking, the move to openness eased our problems with server design, content management, and storage concerns in that now our partners could set up and run their own servers. By creating an open network of distributed servers, we enabled these partners (especially those institutions that use KEEP heavily) to

move their data to locally maintained systems while still being able to contribute their content to a unified, global KEEP collection.

We therefore undertook the work of opening up the KEEP Toolkit through enhancements that involved integrating the toolkit with existing open source repository software, as well as preparing KEEP source code by refactoring the program design and data model to prepare for distribution as open source software.

Integrating An Open Source Repository

By 2004, thousands of KEEP Toolkit users had already created and were currently disseminating multimedia Snapshots. As the collection grew, access to and long-term preservation of content became daunting issues. The initial KEEP Toolkit framework, for instance, did not explicitly offer archival storage and preservation management features. As far as administering the KEEP Toolkit collections, the existing relational database system presented some inherent barriers to maintaining older Snapshots in the collection. Relational databases may perform well for the storage and query of fresh units of information as specified in their current database schema, but they can fall short in the face of even small changes to their data model over time.

Typically, repository systems offer solutions to this common problem, which is why one phase of our move to an open KEEP Toolkit involved integrating DSpace, a standards-based, freely available digital repository system (Smith et al. 2003). This solution of adding DSpace repository connectivity to the KEEP framework is an example of the kind of functional benefits gained by moving into the open source arena. Technically, the move entailed fairly straightforward tasks: creating metadata, packaging KEEP content for export, automating ingest of the content into the repository, developing a preservation strategy, and determining access policies. Moving archived copies of Snapshots into DSpace not only began immediately to solve our preservation needs but also enabled new forms of access to the KEEP content collection. An added benefit of the repository connection is the set of options for sharing content and metadata with other systems which have all been built to standards for finding and sharing digital library content. Because DSpace repositories can be interconnected as digital libraries, the KEEP user community potentially expands to include a broader community of those interested in browsing the archive collections via digital libraries.

Distributing An Open Source Server

As mentioned previously, the KEEP Toolkit database had grown to contain tens of thousands of Snapshots from teachers and students at other institutions. Not surprisingly, KML had also received inquiries from many educational institutions and organizations requesting new features in the KEEP Toolkit software as well as downloadable versions for local server set-up. It became clear that in order to address the requests to expand and adopt the KEEP Toolkit for a variety of institutional purposes fully, we would need to do more than simply purchase a bigger server. In the face of this formidable problem of giving an increasing number of users the customized control and extended functionality they need, an open source approach offers hope. We surmised that by providing opportunities for code customizations and contributions from developers within the community, the open source model could save KML from becoming a bottleneck to future improvements.

Of course, moving to open source development entails more than simply giving away source code. Cultivating a potential community of distributed developers necessitates preparation for issues and concerns at many levels, from code review to project governance. To prepare the server framework code for distribution and eventual extension, the KML analyzed the existing KEEP Toolkit's portability and extendibility. Taking steps in releasing a distributable version required internally reframing existing KEEP software code, testing for system dependencies, scrutinizing the database design, and anticipating where functionality to extend the tool might be needed most. A move was also made towards adding interoperability so that content and metadata mark-up would allow Snapshots to be recognized in a wide range of ways—from integrating with institutional repository systems (as described above) to sharing services with major standards-based e-learning tools and frameworks such as OSP and Sakai.

Initial Results and Future Plans

The effectiveness of the repository and server distribution efforts were finally put to the test in early 2006 with our initial open source release. Within weeks, more than a dozen institutions had downloaded and set up their own KEEP Toolkit servers. Better yet, new feature extensions are already being contributed back to the community as institutions using the distributed framework discover useful ways to integrate with their local systems. As these contributions to the KEEP Toolkit increase, we expect to face important decisions about setting guidelines for community contributions, such as how to incorporate new code into existing lines of development, how to review and test the contributions, and how and when to give outside developers ultimate responsibility over different pieces of the Toolkit code.

Currently, we classify future development of the KEEP Toolkit in two groups: new interior functionality for authoring and new exterior interoperability with outside systems for sharing. For example, interior KEEP Toolkit development has always emphasized visually appealing interfaces for the manipulation of multimedia (e.g., images and streaming video), and we will ensure that this continues to be the case for future variants of the software. Exterior development will continue to focus on interoperability of the KEEP framework content with various other open source systems such as the OSP, topic mapping software such as the Visual Understanding Environment (VUE), and presentation tools such as OpenOffice IMPRESS. Here the notion of a toolkit takes on new meaning as authoring, collaboration, and presentation functionality could be added in a modular fashion through the kinds of interoperability made possible by open source.

Conclusion

The results and lessons to be learned from the early adopters of our open source release are still being gathered and will inform the development steps we take in the future. As we consider what work the KML will undertake and what work our growing community of partners might contribute, we are mindful of the need to focus always on our users (i.e., teachers and learners). The guiding principle throughout the development and release of the KEEP Toolkit has been to promote and sustain the activities that the tool is designed to support. To that end, the technical approach has required looking beyond the functionality of the KEEP Toolkit itself towards the broader issue of interoperability of KEEP content with other open source systems for content creation and sharing. If open source systems truly begin to unify the online environments that teachers use, the KEEP framework will be compelled to provide the means of interoperating with these systems. In the future, the KML will continue to respond to requests from current and prospective users of the KEEP Toolkit for new features and work with the growing community of developers at various institutions to enhance the Toolkit's capacity to accommodate new genres and formats for representing the knowledge of teaching and learning.

References

The Carnegie Foundation for the Advancement of Teaching. CASTL Higher Education. http://www.carnegiefoundation.org/programs/sub.asp?key=21 (accessed September 30, 2006).

Downes, S. 2006. Places to Go: Sakai. *Innovate* 2 (3). http://www.innovateonline.info/index.php?view=article&id=274 (accessed September 30, 2006).

Shulman, L. S. 1986. Those who understand: Knowledge growth in teaching. *Educational Researcher* 16 (2): 4-14.

Smith, M., M. R. Barton, M. J. Bass, G. McClellan, D. Stuve, M. Branschofsky, J. H. Walker, and R. Tansley. 2003. DSpace: An open source dynamic digital repository. *D-Lib Magazine* 9 (1). http://www.dlib.org/dlib/january03/smith/01smith.html (accessed September 30, 2006).

Wheeler, B. C. 2004. The open source parade. *Educause Review* 39 (6): 68-69. http://www.educause.edu/ir/library/pdf/ERM0458.pdf (accessed September 30, 2006).

Looking Toward the Future:
A Case Study of Open Source Software in the Humanities

by Harvey Quamen

The Masters of Arts in Humanities Computing (Huco) program was established at the University of Alberta in 2001. Since then, we have been training young humanities scholars in the intricacies of multimedia, markup languages, project management, research methods, and even game theory (Gouglas et al. 2006). Many of our students—like those in other branches of the humanities—have bright futures in education, law, public relations, marketing, and other disciplines. Increasingly, our students find an outlet for their skills in non-profit organizations, many of which have little to no budget for computer technology or support. One of my goals within the Humanities Computing program has been to introduce open source software (OSS) and its politics of community and sharing into my courses. Open source's wide array of software provides computing solutions for numerous contexts.

When illustrating the advantages of OSS to my students, the exemplar that I cite is a project on which I myself have been working for a few years now—a manuscript-tracking database for *English Studies in Canada* (ESC), a journal on whose staff I serve as an associate editor. The ESC Database, as it has come to be called, is an illustrative case study in that it reveals both the successes and challenges of using OSS in academic contexts that are severely limited by staff and by budget. The project itself is still in progress, but far enough along that a scorecard of successes and failures might be instructive to others in similar situations.

Background: *English Studies in Canada*

English Studies in Canada (ESC) is a medium-sized, peer-reviewed, quarterly academic journal devoted to "any topic which falls into the disciplinary purview of 'English studies' broadly understood" (ESC Digital 2004, "Call for Papers") and is the official journal of the Association of Canadian College and University Teachers of English (ACCUTE).

I joined the editorial team when ESC moved to the University of Alberta in 2003. At that time the team was understaffed and overworked, and the journal itself under-read and overdue. As our first priority, we raised ESC's profile by redesigning the hardcopy ESC. We began to typeset the journal in-house using Adobe InDesign in order to gain control and to save money. We gave ESC a Web presence, and we recruited new subscribers and lobbied for more article submissions—all facets of ESC's public persona.

However, other problems lingered in ESC's private persona. Even though the editorial staff all worked in the same building, communication among us was slow and sometimes inaccurate. Each new manuscript submission generated an overstuffed, legal-sized file folder of documents, but typically only one person at a time accessed the file—and sometimes important files lay trapped in locked offices when editors were off campus or out of town. Hardcopies, communications with authors and with potential peer reviewers, reminders to reviewers to speed up their reviews, inquiries from authors, and the eventual results of the adjudication process itself were not always accessible to the rest of the team, which led to increasingly egregious mistakes: we lost e-mail addresses, some correspondence never made it into the files, and we sometimes forgot that peer reviewers had neglected to send in their reports. I once typeset a supposedly finished article only to be notified by a puzzled author that acceptance of her piece was contingent on making certain revisions that she had not yet even begun.

Our biggest problem was that editors had difficulty finding peer reviewers. Our team scoured our parent organization's membership directory but found the descriptions about member's various areas of expertise to be unhelpful; sometimes the entries were just vague, sometimes they were ambiguous, and sometimes they were downright wrong. The editors begged for a better way to find peer reviewers.

An online database seemed the obvious answer to all of these problems. With such a tool, everyone could get real-time answers to questions and could do so no matter whether one was working on campus, at home, or in the midst of a research trip. Most of us chose not to work in the tiny, cramped ESC office—we used it mostly as a central location for files and other resources—and so a single desktop computer running a database application like Microsoft Access did

not seem to suit our style. Like many other academics, we found ourselves working from our campus offices, from home, from the library, from coffee shops, and elsewhere. In considering how to put our database online for easy Web access, we explored the potential of open source tools; we finally selected MySQL and PHP as the best solution for our staff needs.

MySQL and PHP: Open Source Tools for Variable Web Content

One of the problems facing traditional Web sites coded in HTML is that each Web page becomes static and difficult to change. Software like Macromedia's Dreamweaver (recently acquired by Adobe) or Microsoft's Frontpage can help make Web design a bit more like word processing, and they have helped make Web design less difficult for people who do not wish to deal with specialized code. However, these tools still require ongoing maintenance operations that entail substantial investments in time and personnel. Moreover, the use of standard Web designing software is not a viable option for more complex, dynamic Web sites. The sophisticated search engines on sites such as Google or Amazon, for example, deliver new, unique content for each search one performs; in contrast, if one relies on tools such as Frontpage or Dreamweaver, it is simply not possible to code every possible combination in advance of the off chance that someone might search for those precise results later. Stronger tools are necessary in order to build HTML Web pages out of variable content.

MySQL and PHP are such tools. MySQL, which bills itself as "the world's most popular open source database," can store, organize, and retrieve vast amounts of data stored on a Web server. PHP, a Web-scripting language invented by Rasmus Lerdorf in 1995, acts as the ambidextrous intermediary within such a database environment. On the one hand, PHP can query the database, retrieving records as necessary; on the other hand, PHP can also wrap the data in HTML in order to create Web pages on the fly. By allowing designers to streamline the process of information management while standardizing the process of Web site design, these technologies offer a vital resource for any organization. For academic journals in particular, which must regularly sort, distribute, and update large quantities of information, this open source technology has much to offer. I know first-hand that many other academic journals are in the process of implementing this solution for their own needs; however, for organizations that have not yet taken this step, I believe that an introduction to some of the features of the ESC Database may be of special interest.

The ESC Database: Development and Design

In the summer of 2004, I started using these software tools to write a small Web-based database that would help us track our manuscripts. Our business logic (i.e., the editorial process itself) still was not clear even among ourselves, and so I sometimes relied upon my best guess for handling various situations that had not yet occurred in real life. Yet as occasionally happens, writing the database application served as a way to formalize our own production processes; each major decision about the design of the database essentially entailed a decision about how to structure, distribute, and manage the editorial cycle in such a way that there was an option for every eventuality. For this reason too, the project became larger than I expected and required a greater initial investment of time and energy than I had anticipated. I did not imagine that the project would last any longer than the immediate summer, but I spent the following summer (and nearly every interim holiday) incorporating a fuller range of functions to manage our editorial process. By the time the database finally came online in February 2006, the tiny project had grown to 34 MySQL tables and over 6000 lines of PHP.

While the ESC Database still remains open for future development, the features and functions in the initial version may illustrate the value of MySQL and PHP for other academic journals. First, the database included comprehensive listings of all manuscripts, authors, and reviewers as well the entire membership directory of our parent organization (ACCUTE) and a listing of over 400 department and university addresses. More importantly, the MySQL framework allowed me to organize the Web site to provide convenient searching, navigation, and linking across all different fields of information contained in the database. For example, I was able to incorporate a variety of search techniques based on the major categories of "objects" contained in the database (Exhibit 1). The database design, in turn, allowed any user to establish a range of relationships among different objects by embedding links from one object to the next, which established a cross-referencing network for the search engine (Exhibit 2). This cross-referencing network was further expanded through the keyword functions supported by the MySQL system; the keywords were particularly crucial in helping the

ESC Database fulfill the needs of editorial staff. Users could search for reviewers via keywords that designated particular areas of specialty, search for manuscripts via keywords that designated a given period or focus, or generate lists of potential reviewers for a given manuscript based on shared keywords (Exhibit 3). Moreover, I was able to incorporate a Keyword Thesaurus that permitted both an "exact" mode and a "fuzzy" mode of searching, which gave greater flexibility to users as they searched for manuscripts (Exhibit 4). Needless to say, such capabilities would have been far beyond the scope of any HTML-based Web authoring tool.

The MySQL framework also allowed me to include additional features and functions to support the needs of editorial staff. For example, I designed the database to allow for the creating and archiving of specialized reports to assist editors in their tasks and decisions (Exhibit 5). I also incorporated a variety of administrative pages to provide guidance regarding topics such as HTTP Basic Authentication as well as a function that would allow me to post announcements to users that they would see the next time they logged in (Exhibit 6). In this fashion the database not only provided easy navigation through a complex network of information but also supported timely, convenient communication among members of the editorial staff.

Assessment and Future Plans

Having a no-cost, centrally located means of communication has been essential to our work at ESC. While our operating budget is drawn from subventions from ACCUTE as well as from the Social Sciences and Humanities Research Council of Canada (SSHRC), those contributions cover less than 50% of our annual expenses. We therefore see cutting costs and increasing efficiency as a matter of survival rather than just a means of improving our financial health—and to this end, the ESC database has provided a resource that serves our needs within our budgetary constraints. Meanwhile, the database has also helped provide a technological foundation for our distinctive working environment. Since we all volunteer our time, we work on ESC projects at different times and even at different places. Through the ESC database, editors now can see when new manuscripts enter the editorial cycle, even if they are not communicating face-to-face or sitting in a meeting, and they can all get a detailed account of ESC's workload at any given moment.

While these are significant benefits, the project is still very much a work in progress. The ESB Database is still in a frenzy of data entry; graduate student assistants are combing through our files, aggregating data, and inserting it into the system. After our data entry phase is finished, however, I plan to implement a few changes. First, I underestimated our need for a document management system, and rather than changing the database itself, we are currently considering the possibility of supplementing it with Knowledge Tree—an open source application that will help us keep track of document revisions. Second, the entire database needs a rewrite; this may seem a drastic measure, but it is a necessary one. Even with my best intentions, I have made both coding mistakes and poorly considered decisions in version 1.0 of the ESC Database. For example, creating two separate objects for author and manuscript still feels somewhat inefficient, and the editors have found this aspect of the database a bit counterintuitive. Third, the programming code is voluminous enough that moving to an object-oriented programming style would help simplify the program, reduce errors, and allow new features to be added more easily. Were I to begin again today, in fact, I would opt for object-oriented programming immediately. Open Source advocate Eric Raymond (2001) reminds his readers of Fred Brooks' mantra—"Plan to throw one away; you will, anyhow"—and explains: ". . . you often don't really understand the problem until after the first time you implement a solution . . . So if you want to get it right, be ready to start over at least once" (25).

Such changes to the database would not only improve it for our own purposes; they would also make it a potentially valuable application for open source release. The key factor in this regard is that the database needs to accommodate different editorial procedures and workflows. As its name implies, the ESC Database is tailored to the ESC editorial process. It uses our terminology, embodies our assumptions about how the editorial process works, and takes no account of any procedures that ESC does not implement. For example, ESC has abandoned a traditional editorial decision—"Revise and Resubmit"—in favor of a new category that we prefer: "Accept with Specified Revisions." Any journal using "Revise and Resubmit" in its editorial process has no way to reinsert that category without first reprogramming the entire database, and this feature alone might prevent other journals from using it. Consequently, an open source version of this software will need to formulate and document strategies for its own redesign in different contexts, perhaps by using a variant or subset of RuleML or Business Rule Markup Language (BRML) or perhaps by inventing a new, small, fleet-footed XML-based schema.

In making such adjustments to the ESC Database and eventually offering it as open source software, I believe that it can serve as a prototype for other academic journals to adopt and develop for their own needs. While the staffing policies and editorial processes of academic journals are not all the same, it remains the case that the overwhelming majority of such journals rely substantially on the work of unpaid graduate assistants, interns, and volunteers rather than a cohort of full-time, salaried employees. At the same time, the nature of the work—tracking manuscripts, assigning reviewers, corresponding with reviewers and authors, providing feedback to authors, and preparing manuscripts for eventual publication—is very demanding, and under such circumstances it becomes all the more crucial to ensure convenient access to information while supporting efficient, structured collaboration among all staff members. For journals that have a large base of subscribers and substantial financial support from their hosting institutions, proprietary software applications may already offer the best solution for addressing these challenges. However, for journals that remain constrained by fewer resources, particularly journals in the humanities, open source applications can provide a vital option for achieving the same benefits as they seek to fulfill their own mission.

Conclusion

Since at least the 19th century, the humanities have defined themselves in opposition to the sciences (Mary Shelley's *Frankenstein* is a requisite text in that regard), and academics in this field have become unaccustomed to turning to technology for help. However, the premise of the Humanities Computing program where I teach is that the time has come to tear down that wall and look to the other side to find viable solutions to problems we currently face. Traditional humanists often look for solutions in the past, but it is time to start looking hard at our futures. Open source can both provide our technological solutions and foster our ideological values of collegiality and community, but the onus is currently on humanities scholars to become more computer literate and to educate ourselves about the benefits of open source. That is not only an important lesson I hope to instill in my students, but a lesson that I have already learned in my work at ESC.

References

ESC Digital. 2004. Home page. http://www.arts.ualberta.ca/%7Eesc/ (accessed September 30, 2006).

Gouglas, S., S. Sinclair, O. Ellefson, and S. Sharplin. 2006. Neverwinter Nights in Alberta: Conceptions of narrativity through fantasy role-playing games in a graduate classroom. *Innovate* 2 (3). http://www.innovateonline.info/index.php?view=article&id=172 (accessed September 30, 2006).

Raymond, E. S. 2001. *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary.* Cambridge, MA: O'Reilly. Also available online at http://www.catb.org/%7Eesr/writings/cathedral-bazaar/ (accessed September 30, 2006). [Editor's note: The online version of this source revises and adds to the print source.]

sidebar.

From, By, and For the OSSD: Software Engineering Education Using an Open Source Software Approach

by Kun Huang, Yifei Dong, and Xun Ge

In the discipline of computer science, the disconnect between professional practice and educational preparation often poses a challenge for academic programs. On the one hand, computing is a complex, multidisciplinary field that integrates knowledge from many different areas. This field calls for a workforce of computer engineers and scientists who are able to work in teams in order to apply knowledge and skills from different disciplines in solving problems, to deal with ever-changing situations, and to update knowledge and skills continuously through self-directed learning (McGettrick et al. 2004). On the other hand, most computing education curricula are organized around compartmentalized courses in which students acquire largely incremental, disconnected knowledge and skill sets. Moreover, many computing courses focus more on end products—that is, on the delivery of workable programs—than on the development processes themselves (Upchurch and Sims-Knight 1997). In contrast to the frequent teamwork found in the computing industry, students usually program individually or in small teams at best (Knight, Prey, and Wulf 1997). Consequently, when faced with complex, real-world tasks, students often find it difficult to synthesize and apply their knowledge and skills to solve problems.

Such a focus on self-contained knowledge, individual work, and final products in computing education reflects an epistemological view that "fosters selective inattention to practical competence and professional artistry" (Schon 1983, vii). This paradigm directly contrasts software engineering practice, which is a process of constantly dealing with situated, ill-structured problems and generating viable solutions through individuals' cognitive and metacognitive thinking (Brown and Duguid 1991). Current educational practice lacks just such dynamic knowledge built *in situ*. We maintain that the software engineering profession involves more than software programming and information technologies. It is both an "information ecology" that consists of people, practices, values, and technologies (Nardi and O' Day 1999, 49) as well as a community of practice that people join through legitimate peripheral participation and build their identities through meaning negotiation and interactions with one another (Lave and Wenger 1991; Wenger 1998).

In searching for ways to instill the sense of community through which students learn by solving complex, ill-structured problems collaboratively, we began to examine a special type of software development community—the open source software development (OSSD) community. The OSSD community consists of a group of dedicated, professional developers and user participants who volunteer their expertise and energy in developing open source software over a long period of time toward a common goal. By utilizing online work space and communication tools, these geographically distributed participants collaborate to solve problems through continuous meaning negotiation and knowledge sharing. We believe that the OSSD approach has the potential to address the educational problems discussed above.

Taking a design-based research approach, we have been conducting a series of two studies at the University of Oklahoma (OU), with one study leading into the other. In the fall of 2005, we studied an open source project called Gallery and analyzed why the project was successful from the perspective of collaborative problem-solving, decision-making, and knowledge construction. Based on our initial findings from this study, we then designed and implemented an OSSD environment in a graduate course on software engineering in the spring semester of 2006; during the implementation of the course, we also began to investigate the effects of the OSSD environment on students' motivation and collaborative problem-solving. This article presents the preliminary findings of the first study on an OSSD community, reports the ongoing study on the effects of OSSD on students, and draws useful implications from the two studies. We hope that this research will not only benefit computing education but also education in other disciplines.

From the OSSD: Characteristics Identified

Significant open source products like Apache, Mozilla Firefox, and Linux demonstrate the marked success of the OSSD community. What characterizes an OSSD community is the dedication of time and energy by a group of volunteers in

developing software that is available to use, modify, and redistribute (Hars and Ou 2001). We believe that some OSSD features may be beneficial to current methods of computing education. We therefore selected Gallery, an OSSD community whose size is comparable to that a typical computing class, as the focus of our study. Through careful analysis of the data hosted on the Gallery Web site and other related literature, we have identified several salient characteristics below that contribute to the success of the OSSD community (Ge, Dong, and Huang 2006).

Projects as Shared Enterprise

In an OSSD community, the members' expertise and activities are all geared toward the development of the OSSD project. In other words, the OSSD project drives members' collaboration, knowledge sharing, and learning. If we regard Gallery developers as learners, the starting point of their learning is the Gallery project, and their knowledge and skills are obtained in the process of collaboratively solving complex, real-world problems. Compared with the skills acquired from isolated computing classes, the skills gained from the OSSD development are more holistic and readily applicable to new situations.

Motivation in Joining the OSSD Community

Personal interest motivates OSSD members to join a particular OSSD project. The interest can be in the software program itself or it can arise from the potential of the program to improve the technical skills or marketability of its members (Hars and Ou 2001).

Distributed Expertise and Expanded Community

The knowledge and skills possessed by the OSSD developers are diverse and distributed across the community. Every member has a specialized area. The Gallery team, for example, consists of project managers, developers, database maintainers, graphic designers, user-interface designers, user forum helpers and moderators, Web site maintainers, and translators. The community is not limited to developers; rather, it expands to all kinds of people who contribute to the development of Gallery in various ways. Some may contribute to the source code, others may provide user support, and still others may simply report bugs experienced as users. No matter what expertise the member possesses, everyone can find a legitimate place in the community.

Careful and Systematic Project Planning

Careful and systematic project planning is critical for an OSSD project to ensure the effective collaboration among geographically distributed developers. It gives rise to a roadmap for each of the development goals or milestones, which developers try to reach. In the planning process, the project leader plays an essential role; for example, in the Gallery community, one of the co-founders presents a development roadmap that delineates the project in terms of project features, release schedules, and task distributions. However, the co-founder does not play a dominant role in determining the subsequent steps of the project. Rather, the proposal is subject to the members' scrutiny. Through discussions, the proposal is further defined, clarified, revised, and enriched, ultimately establishing a common ground for members' collaboration.

Collaborative Knowledge Building

OSSD is not only a software development process that leads to a successful and continuously improving product but also a significant learning process for each community member. The Gallery members construct knowledge through multiple rounds of discussions in which they outline problems, share resources and ideas, negotiate solutions, clarify misconceptions, and reach consensus. As a result, each member gains substantial knowledge and experience by working on Gallery.

OSSD System Affordances

In the meantime, the OSSD environment also provides an ideal virtual space that facilitates effective collaboration. The

hierarchical organization of the source code provides a clear visual display for the Gallery developers; the task list decomposes the project tasks into different priority levels and allows the developers to take over tasks and report progress. Different mailing lists allow members to join ongoing discussions and keep updated with any new progress of the software. Bug reports, feature requests, and discussion forums help to establish connections and communications to the wider community, which, in turn, enhances the project improvement.

By the OSSD: Design-Based Research

We believe our findings about the OSSD community are promising, and we see the potentials of implementing its features to address some of the concerns generated by more traditional computing courses. For the spring semester of 2006 at OU, we designed and implemented a computing education course—Computer Science 5213: Software Engineering Processes—with a simulated, full-featured, OSSD environment. Concurrently, we also conducted exploratory research on the cognitive and affective dimensions of the OSSD environment in computing education. The course was a graduate-level course on software engineering processes, and a total of 19 students were enrolled during its first semester. We took an ethnographic approach for the study that used a combination of techniques including conducting observations and interviews and examining student artifacts. The implemented OSSD features are explained below.

Group Forming: Two-Way Selection

The group-forming process was adapted from real OSSD practice but also modified to suit the instructional needs of the course. At the beginning of the semester, the students submitted their resumes to the instructor. Based on the students' experiences and skills, the instructor identified four project founders; interviews were then conducted in the classroom between the founders and the rest of the class in the fashion of a job fair. In these interviews the founders assumed the role of recruiters, and the other students assumed the role of job seekers (Exhibit 1). The process of selection was two-way: Based on their rough ideas for projects, founders could review applicants' resumes and talk with them in order to discover the best match for their needs while the students could shop around to apply for those projects that interested them the most. After the interviews, the founders and job seekers submitted their respective lists of "most-wanted" participants or positions to the instructor, and the instructor tried to match the projects with the students in a way that served the best interests of both parties.

There were at least two benefits for forming groups this way. First, the interview increased the students' self-awareness. The applicants had to think about their knowledge and skills as compared to the project needs and identify the holes in their expertise; in turn, the founders had to reflect on what types of skills and experiences their projects required. Secondly, the four different projects provided the students with selection options, thus promoting their autonomy.

Simulated OSSD and Meaningful Projects

Once the groups were formed, all of the group members discussed the details of their projects based on the ideas initiated by the founders, and then they submitted their project proposals to the instructor. The proposals in this case included plans for a document routing and tracking system (DocBUS), a graphical interface for a local area augmentation system (iLAAS), a Web-based smart event scheduler (SchedulePlus), and an online war memorial (Norman World War II Online Memorial). All of the proposed projects were based on real needs or from real sources, which helped ensure the authenticity of the learning activities. Compared with artificial teacher-assigned projects, these projects were more meaningful and relevant, and the student founders felt more ownership and investment in them. In addition, the selection of project founders ensured that there would be at least one experienced member in each potential group; by working together, the group members could then learn from the valuable experiences shared by their more knowledgeable peers.

Real World Projects as Anchors for Learning

The whole course was organized around the development of the software project. Though the course treated software engineering processes, the students did not learn theories and practice in isolation. Rather, the projects served as anchors

for students to learn software engineering processes by actually following the process of designing and developing real-world software.

For example, as in the real OSSD environment (e.g., Gallery), the students had to plan and document their design and development carefully for each milestone in the process in order to make team collaboration possible (Exhibit 2). These updates were geared toward further enhancing development as well as measuring progress; consequently, the knowledge gained was dynamic rather than inert (Bransford, Brown, and Cocking 1999). Furthermore, such a process avoided a common problem many computing students have—their tendency to go directly to software design and development without sufficient preliminary planning (Upchurch and Sims-Knight 1997). By working on an actual project designed for specific, real-world application, students were provided with a clear foundation that guided their design and problem-solving activities from one stage to the next.

Open Source Thinking and Collaboration

In this course, open source was not limited to the access and use of source codes; more important were the open source thinking and collaboration processes. Since the students communicated through the OSSD system and saved their project documents, meeting minutes, personal reflections, and major decisions in the system, their thinking and collaboration processes became transparent and open to examination, problem-detection, critique, and suggestions. This emphasis on process in the OSSD environment resulted in three key pedagogical advantages for the course as a whole.

First, the open source processes helped to make students' thinking visible and encouraged elaboration and reflection (Linn, Bell, and Hsi 1998). The resulting metacognitive skills are especially important to ill-structured problem-solving, which the students will encounter in their work on a daily basis.

Second, the open source processes also helped to enhance team collaboration. Given the complexity of the real-world projects, it was essential for team members to collaborate effectively on their team project. Since every member was responsible for a project component, which was also connected to other members' components, it was important for the students to see others' work and progress. In this case, the open source approach facilitated the individual members' contribution to the team and the integration of their work into a whole project.

Third, the transparent and visible open source processes allowed the instructor to perform formative assessments constantly as the students developed their projects. The instructor assessed the team progress and the student performance by examining all the logs stored on the OSSD system, including mailing lists, discussion forums, project documents, task lists, source codes, and their contributors. At the same time, students could also log in to the OSSD system to review the work and progress of other individuals or teams, which made them more aware of their own performance.

Instructor as an Expert, Coach, and Project Manager

The instructor plays an important role in the OSSD learning environment, not only as an expert but also as a coach and a project manager. The instructor provides useful advice and suggestions from the perspective of an expert. The instructor also plays the role of a coach, attending to individual students' needs and facilitating students' learning and collaboration processes (Exhibit 3). As an expert and a coach, the instructor detects the bugs during student learning processes, provides the students with in-time diagnostic feedback, and reorients them in the right direction. In addition, the instructor also serves as a project manager who oversees, monitors, and facilitates all the teams and their projects.

Students as Both Developers and Users

In a real OSSD community, members with different levels of knowledge and experience play different or multiple roles, such as developers, testers, or users. This course is designed to encourage all the students to be both developers and users as they participate on two levels of the learning community—the level of the individual team as well as the level of the class as a whole. Within each team, every member works as a developer on a certain module of the software and serves as a user for the other team members in the software development process (Exhibit 4). Moreover, the teams are

also encouraged to follow the extreme programming (XP) model (Beck 2000), which features iterative development and frequent releases. Once a concrete product is released within the class, members of other teams can test it and provide suggestions and feedback as users. The students may download and use the software, detect and report bugs, request additional features, or even fix bugs and provide patches for the other teams. In a traditional course setting, it is usually hard for some students to find ways to contribute, but in this course every member has a place on the team and a role to play within the class as a whole.

In many project-based courses, it is often the case that every team only takes care of its own business and that team members know or care very little about other teams' projects. In this respect, learning in a single context may prevent the transfer of knowledge (Bransford, Brown, and Cocking 1999). However, by comparing their own projects with other teams' projects and by assuming different roles (e.g., developers and users), students are more able to gain substantial understanding of software engineering processes—and, in turn, are better able to transfer their knowledge to new problem-solving situations.

OSSD Virtual Space

The course Web site implemented a similar OSSD virtual working space using GForge, an open source branch of the popular OSSD support platform SourceForge. The support of an OSSD Web site is necessary because of the complexity of the projects and the substantial amount of time required for meetings.

The students were encouraged to utilize the OSSD Web site fully, and their good practice was demonstrated to the class. All four projects were hosted on the same Web site, and each project also had its own home page. The students used this Web site to save and share all their source codes, which were organized into different categories; they also used the Web site to record meeting minutes or major project decisions, plan and allocate to-do tasks, share resources and tools, use mailing lists to communicate with group members, and ask group members or the whole class for suggestions and help (Exhibit 5).

The OSSD Web site thus provided an open record of the student' projects, which allowed students to refer to and discuss their team projects and allowed the instructor to monitor the projects and the team process. The Web site also facilitated student collaboration by allowing them to communicate at any time and to plan, divide, and trace tasks. The use of OSSD tools and the sharing of its resources clearly produce a community knowledge base that is closely tied to the development of software projects.

For the OSSD: Implications

Although our implementation and research are currently in progress, the initial findings and the feedback from the students are promising. Our preliminary data analysis of the Spring 2006 class suggests that students demonstrate a high level of motivation toward their projects, especially in the case of groups that have frequent interactions with real clients. During interviews the students expressed their excitement about having the chance to develop a software project from scratch. In their responses to course evaluation surveys administered at the end of the semester, the students mentioned a number of things that they had learned through working on the real-world projects in the OSSD environment:

- They learned how to find a position and play a role in their teams in order to make unique contributions to a large-scale project.
- They gained knowledge based on project needs, and they learned from more knowledgeable peers.
- They learned to use the OSSD virtual space to manage large-scale projects with which they had little previous experience.
- They felt more capable of working with different people.

Furthermore, we found that factors such as project authenticity level, instructor intervention, and project founders' leadership had a great influence on the success of the classroom OSSD projects. More successful projects such as DocBUS moved beyond the scope of the course as some members continued to work on the project even after the course

was finished.

As our design-based research continues, we hope to identify special characteristics of OSSD in the classroom setting and to gain a deeper understanding of student cognition, motivation, and behavior in the OSSD environment. We also hope to generate instructional strategies for implementing successful OSSD in the classroom setting. In the future we look forward to building a project database and forming partnerships with other computing programs, universities, industries, or companies. In addition, we would also like to extend the OSSD concept and approach to other disciplines such as math, science, and engineering.

Conclusion

We hope that the OSSD approach provides an alternative instructional approach to address the issues and challenges faced in computing education. Ideally, computing education nurtures graduates who are competent in domain knowledge, skilled in complex problem-solving, and capable of self-directed learning. We expect that our future research will shed more light on the solution, so that what we as educators learn from the OSSD approach—both the process and the product—will allow us to contribute back to a larger computing community by training highly qualified software engineers.

References

Beck, K. 2000. *Extreme programming explained*. Boston: Addison-Wesley.

Bransford, J. D., A. L. Brown, and R. R. Cocking. 1999. *How people learn: Brain, mind, experience, and school*. Washington, DC: National Academy.

Brown, J. S., and P. Duguid. 1991. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization Science* 2:40-57.

Ge, X., Y. Dong, and K. Huang. 2006. Shared knowledge construction process in an open source software development community: An investigation of the Gallery Community. Paper presented at the 7th International Conference of Learning Sciences, Bloomington, IN, June-July.

Hars, A., and S. Ou. 2001. Working for free?—motivations of participating in open source projects. Paper presented at the 34th Hawaii International Conference on System Sciences, Outrigger Wailea Resort, Hawaii, January. http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf (accessed September 30, 2006).

Knight, J. C., J. C. Prey, and W. A. Wulf. 1997. Undergraduate computer science education: A new curriculum philosophy & overview. Paper presented at the 1997 Frontiers in Education Conference, Pittsburgh, PA, November. http://www2.umassd.edu/swpi/UVa/curriculum.pdf (accessed September 30, 2006).

Lave, J., and E. Wenger. 1991. *Situated learning: legitimate peripheral participation*. Cambridge: Cambridge University Press.

Linn, M. C., P. Bell, and S. Hsi. 1998. Using the Internet to enhance student understanding of science: The knowledge integration environment. *Interactive Learning Environments* 6 (1/2): 4-38.

McGettrick, A., R. Boyle, R. Ibbett, J. Lloyd, G. Lovegrove, and K. Mander. 2004. *Grand challenges in computing education*. Swindon, UK: The British Computer Society.

Nardi, B. A., and V. L. O'Day. 1999. *Information ecologies: Using technology with heart*. Cambridge, MA: The MIT Press.

Schon, D. A. 1983. *The reflective practitioner*. New York: Basic Books, Inc.

Upchurch, R. L., and J. E. Sims-Knight. 1997. Designing process-based software curriculum. Paper presented at the Tenth Conference on Software Education and Training, Virginia Beach, VA.

Wenger, E. 1998. *Communities of practice: learning, meaning, and identity*. Cambridge, MA: Cambridge University Press.

Places to Go: Intute

by Stephen Downes

"A long wait," writes Graham Attwell. "But worth it" (2006, ¶ 3).

Attwell is reacting to the July 13, 2006 launch of Intute, an online service that provides free access to a wide range of well-selected Web resources for educators and educational researchers. Created by a network of British universities and the Joint Information Services Committee (JISC), a federal British initiative, Intute represents what learning object repositories were meant to be but never became, as Attwell's remark attests. As noted by the Intute site managers, "Automation can't replace our human value judgments, nor can it be responsive and dynamic to meet the real needs of UK HE [higher education] and FE [further education]—but Intute can . . . You can explore and discover trusted information, assured that it has been evaluated by specialists for its quality and relevance" (2006, ¶ 4, ¶10). This claim is borne out in the design and features of the Intute repository, a project that signifies a decisive step forward in the online dissemination of educational resources and scholarship.

The Intute Web site's apparent simplicity belies its complex construction, but an exploration of the home page is the best way to begin. A visitor will first notice Intute's four major subdivisions—Science and Technology, Arts and Humanities, Social Sciences, and Health and Life Sciences—presented as four major panels across the top of the page, each clearly color-coded.

Entering one of the sections—perhaps the "green" one, Social Sciences—leads to a simple page that allows the visitor to search or browse by headings. A navigation menu appears in the left column, with news and features appearing in the right column. Readers will find this layout immediately familiar; it is typical of sites across the Internet. New visitors will dive right into the center column while repeat visitors will focus on the news at the right.

A reader may click on one of the subheadings, perhaps Economics, and be greeted with a further list of subheadings. At the bottom of this list, a new item appears; in this case, the page provides a notation that reads "Editors: University of Bristol, University of Newcastle upon Tyne." Following the link to this notation reveals the first major feature of Intute: information about the editors who select the best material from a given discipline.

Clicking on one of the further subheadings, such as Macroeconomics, brings the reader to the first set of resources, a list of further subheadings, and a list of related sections. As always, the search function is available, as is a thesaurus. Each resource listed includes a title link that takes you directly to the resource online, a link to the full record via the details icon, and a bookmarking tool via the checkmark icon. The full record displays some basic information (also known as metadata) about the resource: when and where it was published, who wrote it, what type of resource it is, what format it is in, how it is classified, and more.

At this point the reader may be thinking that this is just another repository. The resources are invariably offsite, located at universities and agencies around the world, but there is nothing unusual in this. The resources are screened by qualified editors. But again, this is not that unusual; About.com has been using human editors for years, and in the academic world, professors have been informally collecting lists of recommended resources since before the Web was born. Click, though, on the A to Z of Services link from the main page or any of the subpages, and a very different picture begins to emerge.

The main site's list of services is almost too overwhelming; it is usually easier to focus on a particular section, such as the Social Sciences List of Services. And the first item that greets the reader on this page is a seemingly unacademic service: a blog. The Social Sciences Blog lists news and events from the social sciences, new entries in the repository, and more, depending on the interests of the writers. Though the blogs are not sufficiently detailed (we would like to see blogs within subcategories—for example, "Economics" or even "Macroeconomics"), the service nonetheless puts a human face on the otherwise sterile repository.

Some of the other services become immediately apparent. The Conferences and Events page, for example, is exactly as

advertised (though it does impose an additional step to locate conferences rather than immediately displaying a list). The ePrints service links to a list of Intute projects produced in collaboration with other institutions; in this case, the objective is to make institutional listings of academic journal publications (or ePrints) accessible to Intute's search service.

One of Intute's major innovations—and the service that separates it from previous repositories—is the Harvester. This service is an automated computer program that contacts remote databases and retrieves records of newly created resources. The records contain the basic information, or metadata, of these resources, which is then stored in Intute's database. The resources themselves remain where they are, scattered across the Web, not transferred until they are needed.

What makes this possible is that many other repositories and content services have begun to list their resources in XML. For example, services using Open Archives Initiative (OAI) (cf. Downes 2003) typically provide metadata in the Dublin Core metadata format. More complicated metadata, such as the Metadata Encoding and Transmission Standard (METS), is used by more specialized services. Other repositories, such as MERLOT and CAREO, make metadata available in RSS. The growing use of such XML-based metadata formats is significant because it means that a service such as Intute can index all these resources automatically.

How this directly helps the reader is easily evident in the search function. A standard search will reveal resources reviewed and selected by Intute's editors. A search using the Harvester, however, reveals many more resources collected from the many harvested repositories. Thus, Intute combines the precision of human editing with the scope of a harvest-based search.

Just as Intute gathers data from services around the Web, it also provides data to other services. The Working With Intute page provides advice for developers wishing to include Intute data on their own Web sites or with their own services. For example, Embedding Intute provides details of the RSS and OAI feeds that may be accessed by other services. Alternatively, developers can place an Intute search in their Web site. Additional metadata services are being investigated as well.

What Intute represents is a turning point in the sharing of academic resources. Previous efforts to develop repositories, including learning object repositories, have adopted the federated system whereby repositories are linked into what amounts to a closed network with access to resources protected via a common network login (such as provided by Shibboleth). The network search, meanwhile, would scan each repository, one by one, any time a search was made. This approach protected proprietary resources that may have been owned by another institution, but it did so at the expense of making the search very slow; simultaneously, it excluded many materials from the wider Web that would have been of interest to educators.

Intute, therefore, is the first of a new generation of repositories, creating loose associations of resource providers, not through a secure and tightly connected network but rather through an open and shared collection of metadata resources. The academic network is therefore designed along the same model as the wider world of blogs, online news articles, and content management systems—and can both feed into and collect resources from that wider world. In this way Intute not only helps teachers, but it also helps students (whether enrolled in an institution or learning informally at home) and other service providers worldwide—all at no cost.

Intute software has not yet been released; however, Intute Executive Director Carol Williams writes that "UK repository search service work (delivered in partnership with UKOLN and SHERPA) is about to start its next phase and the intention is that we take a proactive approach to dissemination," which will include the provision of open source software (C. Williams, personal e-mail, August 1, 2006). It is difficult to imagine what an interconnected network of sites such as Intute will look like—but the prospect is an exciting one.

References

Atwell, G. 2006. Intute—The right way to provide educational resources. Weblog entry, July 26. The Wales-Wide Web.

http://www.knownet.com/writing/weblogs/Graham_Attwell/entries/9257243250 (accessed September 30, 2006).

Downes, S. 2003. Open Archives Initiative. *The Technology Source* (May/June). http://technologysource.org/article/open_archives_initiative/ (accessed September 30, 2006).

Intute. 2006. About Intute. http://www.intute.ac.uk/about.html (accessed September 30, 2006).