Adopting Open-Source Software Applications in U.S. Higher Education: A Cross-Disciplinary Review of the Literature
Shahron Williams van Rooij

REVIEW OF EDUCATIONAL RESEARCH 2009; 79; 682 originally published online
Feb 18, 2009;
DOI: 10.3102/0034654308325691

The online version of this article can be found at:
http://rer.sagepub.com/cgi/content/abstract/79/2/682

Published on behalf of

American Educational Research Association

By

SAGE

Additional services and information for Review of Educational Research can be found at:

Email Alerts: http://rer.aera.net/cgi/alerts
Subscriptions: http://rer.aera.net/subscriptions
Reprints: http://www.aera.net/reprints
Permissions: http://www.aera.net/permissions
Adopting Open-Source Software Applications in U.S. Higher Education: A Cross-Disciplinary Review of the Literature

Shahron Williams van Rooij
George Mason University

Higher Education institutions in the United States are considering Open Source software applications such as the Moodle and Sakai course management systems and the Kuali financial system to build integrated learning environments that serve both academic and administrative needs. Open Source is presumed to be more flexible and less costly than commercial software. This article reviews the literature from the fields of Software Engineering and Education to determine the state of the current body of knowledge around the key drivers of Open Source adoption. The author discusses gaps in the literature and identifies opportunities for more rigorous research to measure the effectiveness of Open Source software in creating a balance between sound pedagogy and business efficiencies.

KEYWORDS: open source, higher education, technology, learning environments, decision-making, instructional technology, administrative technology.

Open-source software—software delivered with its source code—is an outcome of the convergence of information and communication technologies (ICTs). Ubiquitous computing that includes high-speed, high-memory hardware; embedded and invisible devices; enterprise-wide networked software; voice-activated systems; and personal data warehousing, not to mention the growth of high-speed Internet access, has contributed to that convergence. The proliferation of small, multifunctional devices (phones that take digital pictures, send e-mail, etc.) is a physical manifestation and, depending on one’s perspective, a benefit of that convergence. More important, the convergence is permanent; working, living, and learning will never be the same as before this convergence (Stalling, 2000).

For institutions of higher education in the United States, ICT convergence encompasses the convergence of the traditional technology of the university campus—administrative systems such as student information, finance, human resources, and fund-raising systems—with the emerging technologies of online, flexible learning (Taylor, 2003). Once academic and administrative software applications were moved to new high-speed, high-capacity networks, it was initially a no-brainer to provide faculty and staff members and students with the tools and
Adopting Open-Source Software

training they needed to travel the information superhighway around the world (Hengehold, 2001).

However, this convergence has not been painless. From the perspective of institutions’ technology staff members, the growing demand for support of distributed learning is one of the top 10 issues consuming their time (EDUCAUSE Current Issues Committee, 2006) but deemed essential for institutions’ strategic success. From the point of view of faculty members, opportunities for development, support, training, and some form of compensation for using ICT are in demand. Last but certainly not least are students, the majority of whom are employed, not dependent on their parents, and seek engagement as well as knowledge (National Center for Education Statistics, 2004). All of this is taking place in a climate in which traditional funding sources have become less generous, regulatory requirements are becoming more stringent, and accountability has become the battle cry.

In response to these trends, there is some indication that institutions are seeking to reduce the technical support crisis by increasing efficiencies. One school of thought contends that open-source software will provide both faculty members and the technology staff members who support them with enough flexibility to maintain the correct balance between technology and pedagogy (Pavlicek, 2000; Weber, 2004; Williams, 2002), enabling the construction of integrated learning environments that serve both the academic and administrative needs of institutions. This school of thought points to the successful rollout of open-source software products such as Linux and Apache over the past 10 years to support claims of the long-term viability of open-source software as an alternative to commercial software products. The problem, however, is that these open-source products were developed for technologists, not for educators or their students.

The chief information officers (CIOs) and chief financial officers (CFOs) of large research institutions have been among the most vocal proponents of open-source software. For example, while employed at a higher education software and services development company, I experienced this evangelism firsthand during the 2005 conference of the National Association of College and University Business Officers (NACUBO) held in July in Baltimore. At a preconference workshop on the Kuali project, an open-source financial system being developed by a group of six universities with funding from the Andrew W. Mellon Foundation and endorsed by NACUBO, a joint panel of CIOs and CFOs encouraged commercial vendors to provide implementation and support services to colleges and universities seeking to adopt Kuali software modules, particularly services that would enable the integration of open-source and proprietary software systems. The panel cited the value of open-source software as being greater technology efficiency and lower costs for institutions. The CIOs also mentioned their involvement with Sakai, an open-source platform to construct comprehensive learning management systems, and the growing popularity of Moodle, an open-source course management system developed in Australia. However, no mention was made of the nontechnical, administrative users of the Kuali financial system or of the faculty and student users of Sakai and Moodle. Technology and economics appeared to be driving (or trying to drive) decisions about administrative and academic software applications, not functionality or, in the case of academic software, pedagogy.

The following questions frame this review:
What are the key drivers underlying the adoption of open-source software as a viable means of capitalizing on ITC convergence to serve both the pedagogical and business needs of U.S. institutions?

What direction should educational researchers pursue to assess the effectiveness of open-source software applications in building an integrated learning environment for the academic and administrative sides of institutions?

The review begins with the origins and definition of open-source software and what differentiates open-source applications for technical versus nontechnical users. The method used to search, synthesize, and analyze the literature is described, followed by the identification and critique of the themes around the key drivers of open-source software adoption in the software engineering and education literatures, in the context of satisfying academic and administrative needs. The review concludes with opportunities for future research of open-source software’s effectiveness in capitalizing on ITC convergence to achieve a balance between technology and pedagogy, particularly for U.S. colleges and universities.

Definition of Open-Source Software

The underlying concept of open-source software is access to the human-readable programming statements created in a programming language such as Java or C++, so that the source code can be used and/or modified by anyone, usually under the terms of a license agreement (President’s Information Technology Advisory Committee, 2000). With origins in the Free Software Foundation (FSF; Stallman, 1999), the concept has expanded beyond the availability of source code to the concept of the freedom to run, modify, and distribute copies of a program, either free of charge or for a fee. The Open Source Initiative (OSI), an organization dedicated to managing the open-source campaign and its certification mark, specifies what is permissible in a software license for that software to be referred to as open source (Perens, 1999).

The FSF and the OSI (n.d.) represent two distinct philosophies in the open-source movement. The FSF calls for the freedom to (a) run a software program for any purpose, (b) study how the program works, (c) adapt it to an individual’s or organization’s needs, (d) redistribute copies to help other developers, and (e) improve the program and release those improvements to the whole community. The OSI espouses free redistribution, including source code, the distribution of derived works, no discrimination against persons or groups as contributors, no discrimination against fields of endeavor, and licensing that does not restrict other software. Common to both the FSF and the OSI is the belief in access to source code. However, the FSF believes it to be immoral and unethical to use anything other than free software, whereas the OSI believes that there is a place for both open- and closed-source software. It is the OSI’s concept of open source that I use throughout this review.

Open-source software products available today are primarily infrastructure products. Figure 1 provides a visual representation of the technical infrastructure of networked computer systems on the basis of the taxonomy of Kurose and Ross (2004), along with some of the corresponding commercial and open-source software applications. A data network is a four-layered stack, with each layer having different functions and protocols. At the bottom of the stack is the hardware, which...
includes the server boxes, network switches, hubs, and bridges that are the physical components. Only commercial vendors operate in this layer, with firms such as Hewlett-Packard, Dell, IBM, and Cisco among the many providers. The next layer up is the network layer, which formats data and manages the movement of data round the network. Linux, the open-source operating system developed by Linus Torvalds in 1991 and now the operating system with the largest number of users worldwide ("Linux Distributions," n.d.); the Apache operating system for Web servers; and the Darwin operating system for Macs are examples of open-source software at the network layer. Commercial systems at this layer are UNIX and Windows. The transport layer manages the exchange of data from one computer to another and is the layer in which encryption programs are normally placed. OpenSSL is an open-source encryption program, whereas McAfee produces a commercial encryption program. Common to the hardware, network, and transport layers is the fact that they all serve technologists, and the users of the infrastructure software are also technologists.

At the top of the stack is the application layer, where the programs that provide specific functionality to nontechnical users reside. User interfaces, such as Web browsers or desktop windows, enable nontechnical users to interact with software at the application layer. Open-source course management systems such as Moodle (About Moodle, n.d.) and Sakai (The Sakai Foundation, n.d.), as well as the Kuali open-source financial system (The Kuali Foundation, n.d.) currently in development,
Williams van Rooij

are at the applications layer of the technology stack. Examples of commercial systems at this layer are Blackboard, Desire2Learn, Angel, and Oracle Financials. Because the users of the software at the applications layer are functional, (typically) nontechnical users, the software at this layer, whether open source or commercial, is a supporter or enabler rather than a driver of an organizational process and as such must meet functional needs first and foremost. Consequently, when evaluating open-source versus commercial course management systems, for example, the balance of technical efficiency and pedagogical need must be considered.

This review focuses only on open-source software at the application layer, with emphasis on systems that support teaching and learning.

Method

Scope

Using the portal of the Association for Computing Machinery (ACM), as well as electronic database searches of the Education Resources Information Center, ScienceDirect, Dissertation Abstracts International in the fields of engineering and education, and the EDUCAUSE Center for Applied Research, a search of English-language books, book chapters, peer-reviewed journals, and professional associations with well-established peer-review processes for conference papers and reports was conducted, with the same keywords used for all searches. The searches returned a total of 4,882 unduplicated references, the majority (63%) through the ACM portal. The scope of the search was then limited to benefit/risk theories and critical success factors for the adoption and deployment of open-source software, which excluded 2,644 of the references. Finally, 2,180 references focused on open-source software at the network infrastructure level rather than the nontechnical application user level where course management systems fall, leaving a total of 58 resources to be included in this review.

Process

Each document was read, notes were taken on key points, and the notes were uploaded into NVivo, a software program for the management and analysis of text data. Using the software’s visual coding capabilities, words, phrases, and activities that seemed to be similar were grouped into categories or themes. Emergent themes were identified and labeled to serve as the foundation for preliminary analysis. The software enabled the construction of an audit trail (Hoepfl, 1997) or schema that mapped the themes with their authors and context, so that multiple perspectives on a given theme could be captured and identified. Five distinct NVivo nodes or themes emerged, and I discussed the themes with software developer and systems analyst colleagues at my technology organization and with academic and administrative computing professionals at higher education conferences to ensure that they made sense to subject matter experts. The choice of narrative analysis of the literature instead of meta-analysis was based on the facts that (a) only 6 of the 58 resources used systematic data collection with reported sample sizes and statistical analysis, and (b) the remainder focused on adoption from such varied perspectives as cost perceptions, the dynamics of collaboration, and antivendor sentiment, as to preclude meaningful comparisons.
Findings

The first question framing this review concerned the identification of the key drivers of open-source software adoption. Five themes dominate the literature: (a) social and philosophical benefits, (b) software development methodology benefits, (c) security and risk management benefits, (d) software adoption life cycle benefits, and (e) total cost of ownership benefits. All provide perspectives about the relationship between open-source software and closed-source software and the relative merits of each. All five themes have also been used as arguments for and against the use of open-source software in U.S. higher education. However, all focus on open-source software from the perspective of technologists, not educators.

Social and Philosophical Benefits

Analyses of open-source software using social science theories (Ciffolilli, 2004; Coleman, 2004; Franck & Jungwirth, 2003; Glass, 2003; James, n.d.; Lessig, 2002; O’Mahoney, 2002; Perens, 1999; Stallman, 1999) focus on stripping away the economic factor from technology and adopting open-source software for the common good. Software licenses are perceived as an injustice, the drive for sales revenue and employee compensation deemed to contribute to privatization and the depletion of public domain knowledge. The social movement theorists often cite Microsoft as the embodiment of the injustice of commercial software and the profit motive, even calling for government support on the basis of the argument that free access will bring down the cost of information and that price at marginal costs is consistent with neoclassical economics under which government operates.

Using the case study method, Eduventures analysts Burdt and Bassett (2005) investigated the motivations and decision-making rationales of eight senior information technology (IT) administrators. Study participants mentioned cultural fit between open source as a social movement and public education as one of the reasons for institutions of higher education to explore open-source software. Similarly, more than half (58%) of the 257 higher education CFOs participating in a survey sponsored by NACUBO stated that the freedom to modify software source code was the primary reason for their interest in adopting open-source software applications (Hignite, 2004).

The analysis of open source as a social movement has taken on an almost religious zeal, with true believers pushing open-source software as the epitome of technology for the common good. This appeal to the common good would certainly resonate in higher education, in which such values are well entrenched and indeed part of the mission of higher education. As such, it may offer some ideological or emotional motivators for higher education to examine open-source software. However, it provides no insights into how or how well, at the practical level, either the teaching and learning needs or the administrative needs of an educational organization can be addressed. It focuses only on the needs of technologist-developers, who believe in freedom of access to software source code, and offers no empirically measurable evidence as to the effectiveness of open-source software in capitalizing on the convergence of academic and administrative technologies.

Software Development Methodology Benefits

In characterizing open source as a new development paradigm, the primary focus is on the decentralized, distributed nature of development, with members
who come and go, and where community-wide testing and debugging of the software is an ongoing process (Barley, 1990; Behlendorf, 1999; Evans, 2002; Franck & Jungwirth, 2003; Gacek, Lawrie, & Arief, n.d.; Kelty, 2004; Rossi, 2004; Rothfuss, 2002; Von Krogh, 2003). Raymond (2001) identified two distinct approaches to software development that he called the “cathedral” model and the “bazaar” model. In the cathedral model, which is the traditional model of software development, projects are managed centrally under strict timelines and staffing, and beta testing and quality control protocols are clearly defined and documented. In the bazaar model, which is the model for open-source development, authorship is decentralized, projects have distributed team members who come and go, and community-wide testing and debugging of software is ongoing. Raymond likened participation in open-source development without financial compensation to the sociological or anthropological concept of a gift culture, in which the only available measure of competitive success is reputation among one’s peers.

In a multiyear comparative study of four software development communities, Scacchi (2001) referred to the open-source requirements process as a sociotechnical process that includes the development of constructive social relationships, informally negotiated social agreements, and a commitment to participate through sustained contribution, which enables open-source communities to function without central corporate authority. Stewart and Gosain (2004) developed a framework of the open-source ideology and a theoretical model to show how adherence to components of the ideology affect the effectiveness of open-source development teams. The model is based on the idea that ideology provides clan control, which is important in open-source development settings because open-source development teams lack formal behavioral and outcome controls. The authors then conducted a survey of project administrators and observation data from 48 selected open-source projects listed on SourceForge. The results of the study affirmed Stewart and Gosain’s hypotheses, although the relative importance of norms, beliefs, and values varied in different types of team effectiveness.

For Weber (2004), participation without monetary compensation is motivated by the desire for recognition by one’s own peers, by the drive for personal satisfaction, and by the opportunity to perfect one’s own experience. David, Waterman, and Arora (2003) also explored the motivation for voluntary contribution to open-source projects. In a survey of 1,588 free, libre, and open-source software developers in Western Europe and North America, the primary reasons given were community participation (which 77.8% rated as very important or important), promotion of the open-source method of development (68.6%), promotion of the ideal of freely modifiable software (78.6%), and providing an alternative to proprietary software (61.9%).

In a keynote presentation at the 2004 EDUCAUSE conference, H. D. Lambert (2004), vice president and CIO of George Washington University, defined open source as producer universities and (possibly) vendors working together with stakeholders on innovative software that fulfills academic priorities. Lambert (2005) later proposed that institutions focus on the scholarly information systems in which open-source software already has a foothold (portfolios, course management systems, etc.), while working to open up the student information systems environment using collaborative development and open-source standards. Lambert warned, however, that higher education still needs to work with the vendor
Adopting Open-Source Software

community to minimize risk and must establish a new organizational vehicle for the community to address challenges and barriers.

In the open-source software development paradigm, developers are also the users of software. Thus, it makes sense for developers to see benefit in participating in the collaborative development process. In higher education, however, business applications, including administrative systems and course management systems, are used by individuals who may have varying levels of technical expertise. Consequently, looking at open source from the perspective of software development methodology offers little insight about serving both the academic and the administrative needs of an institution of higher education.

Security and Risk Management Benefits

One of the tenets of open source philosophy is that open-source code is more secure, because vulnerabilities are easier for users to find and fix, and “many eyes” are examining the software (Raymond, 2001; Stallman, 1999; Weber, 2004). However, it is not yet clear what impact, if any, open source has on software security. Anderson (2002) developed a mathematical analysis that showed open-source and commercial off-the-shelf software to be equally secure, noting that what is important is how one defines “security” and what a secure system should (not) do. Collins and Mansell (2004) stated that a critical weakness in discussing security concerns current methods and procedures for identity verification and authentication and what constitutes an “acceptable” level of failure of these systems. Koetzle et al. (2004) noted that IT security experts at multibillion dollar companies choose open-source software to cut costs, not because of a belief in the inherent security of open-source software. Schneider (2000) contended that the tenets of open-source development are incompatible with business models that depend on successfully promoting secure systems, while Viega (2004) suggested that those who do look at code for security problems are either the “altruistic” types seeking a safer world or those trying to promote themselves or their companies.

Wilson (2004) stated that the community-managed nature of open-source projects does not result in any perceptible difference in security compared with commercial or proprietary software. The relationship between development methodology and security was echoed by Obasanjo (2002). Even open-source guru Raymond (2001) stated that security challenges are grounded in bad design, not source code access. Neumann (2000) noted that what is required to achieve secure systems is a disciplined approach to the software development process.

Nevertheless, the security issue remains a barrier to the widespread adoption of open-source software (Lawton, 2002; Woody, Hall, & Clark, 2004). Mobility, ubiquity, and ease of use are polar opposites of security, with no well-defined way of addressing the trade-offs. Both open- and closed-source development are focused on “making systems do things,” not on keeping systems secure.

It is clear from these discussions that security is an issue that goes well beyond open source versus closed source. What is also clear is that the security at any level is dependent on the human factor: the skill, knowledge, discipline, and vigilance of developers and administrators in building and managing software. For U.S. colleges and universities, where IT staffing and funding are chronic challenges, recruitment and retention of the talent required to develop and manage secure open-source applications may be a much bigger hurdle than is currently believed.
Software Adoption Life Cycle Benefits

The issue of adoption harkens back to the social sciences theory of how and why new ideas spread through cultures. Rogers (1995) first introduced his diffusion of innovations theory in 1962, grouping innovations into categories (innovators, early adopters, early majority, late majority, and laggards), with each adopter’s willingness and ability to adopt an idea or innovation dependent on that individual’s awareness, interest, evaluation, trial, and use of the innovation. Diffusion theory has been applied to the adoption of technology by businesses and individuals. In Crossing the Chasm, Geoffrey Moore (1991) adapted Rogers’s categories to the challenge of marketing technology products, particularly those technologies not within the realm of existing values and past experiences. In a presentation at the Open Source Business Conference, Moore (2005) updated his adoption life cycle to assess how open-source development can continue to help large organizations, particularly those in the technology industry, differentiate themselves successfully over the adoption life cycle. Moore sees specific open-source projects at different points in the life cycle. Linux, for example, is in the “early main street” phase.

Other adoption models focus on costs and the trade-offs between product features, risk, and cost in technology adoption (Dedrick & West, 2003; James, n.d.; Kwan & West, 2005; Overby, Bharadwaj, & Bharadwah, n.d.). In 1995, Gartner Group analyst Jackie Fenn (Linden & Fenn, 2003) took the technology adoption life cycle to the next level and developed what the Gartner Group calls the “hype cycle” to characterize the overenthusiasm and subsequent disappointment that typically happens with the introduction of new technologies. Evans (2002) also looked at open source in the context of the hype cycle, stating that the hype surrounding open source is ahead of the reality. Like the Gartner Group, Au and Kauffman (2003) believed that the trend for most new and emerging technologies is to go through the hype cycle. Decision makers more averse to risk are more likely to make decisions later, when enough information has been available and processed properly. As such, the adoption of the more proven open-source software products (e.g., Apache, Linux) is more likely than that of newer, unproven open-source business applications.

The adoption literature recognizes organizational predisposition to open source and the technology skills of staff members as critical to open-source software adoption. At the bottom layers of the technology stack (operating systems, developer tools, etc.), there are professional services organizations, such as Red Hat and IBM, that provide support and training to assist organizations seeking to adopt open-source operating systems and other infrastructure tools but that may not have staff members with the necessary know-how to make open-source adoption a success. Moreover, these professional services organizations are providing their services directly to the technical users of these systems and tools. For academic and business applications, these types of services are almost nonexistent. Moreover, it is also unclear as to whether institutions of higher education could afford to contract for these services if they were available.

Total Cost of Ownership Benefits

Where the literature on open-source adoption is less robust than the areas previously reviewed is in the area of cost of ownership. Because there are no license fees
for the source code, the underlying assumption is that open-source software is less expensive to use than commercial software (Carmichael & Honour, 2002; Hahn, 2002; Hirst, 2001; Jasinski, 2004; Wheeler, 2003; Williams, 2002). To date, however, there is little systematic research to support this assumption. What does exist are beliefs, attitudes, and opinions of higher education technologists responding to the latest wave of state cuts in higher education funding that began in 2001 and changing software vendor business models, creating fear, uncertainty, and doubt about commercial software. Stunden (2003) sees open-source adoption as higher education’s response to the flurry of vendor mergers and acquisitions. In the past 5 years, for example, the field of vendors selling administrative software applications to higher education in the United States has declined from about a dozen to three main players—SunGard Higher Education, Datatel, and Oracle/PeopleSoft—and a few niche players focused on small institutions with limited funds and needs. Similarly, the major vendors of course management systems, Blackboard/WebCT, have shifted their business models to enterprise-wide functionality, resulting in price increases for both licenses and maintenance as much as 200% above 2001 levels. In October 2005, a Blackboard buyout of WebCT placed the long-term future of WebCT’s products in doubt (Carnevale, 2005). Stunden’s own institution, the University of Wisconsin—Madison, has a large technical staff that could develop administrative and academic software applications in house. However, Stunden noted that the real challenge is implementation, support, and maintenance. To be successful with open-source software requires collaborative implementation, support, and maintenance, which are not yet part of the higher education culture, according to Stunden.

There is some evidence that the absence of vendor license fees may not translate to real savings for institutions. Moore (2002) warned institutions that managing open-source courseware and knowledgeware development can be almost as labor intensive and expensive as buying a proprietary product and adjusting it to fit a particular institutional culture. Speaking to college and university technologists at the eGovOS Conference, Alterman (2004) stated that open source is actually a marketing strategy by which vendors make money by selling support and other services to institutions adopting open-source software. Only the top-tier institutions of higher education—the doctoral and research Carnegie 200—are rich enough in IT support, including graduate students and postdocs, to go it alone with open-source software or contract for vendor support services should they choose to do so. Similarly, Green (2004) noted that the greatest support for open-source software is among research universities. Even in that segment, however, fewer than one third agree that open-source software is a viable alternative to their vendor-supported administrative systems, such as student records, human resources, and so on. However, open-source software is perceived as a viable alternative for course management systems. These results remain relatively unchanged in Green’s (2005) later survey. Moreover, more than half (56.8%) reported that the selection of course management decisions is made by IT professionals. Only at community colleges was there strong collaboration (44.3%) between the IT and academic departments on course management system selection decision making. This suggests that technology drives pedagogy and that teaching and learning must adapt to the learning environment that “fits” an institution’s technology goals.
To date, evidence of the cost advantages to higher education of open-source versus commercial software has been primarily anecdotal. For instance, in the July 2006 issue of the trade publication *Campus Technology*, the CIO of a private master’s institution with a Higher Education Publications directory (HEP) enrollment of 3,500 purports to have saved 20% in annual maintenance costs by switching from vendor products to open-source software (Villano, 2006). What is not stated is what type of open-source software (infrastructure level vs. application level) the institution uses or the number and skill set of the institution’s programmers. In a recent survey of 195 CIOs conducted by the IMS Global Learning Consortium (2006), an international educational technology standards organization, more than half (57%) of the respondents purported to have seen improvements in cost of ownership since replacing vendor products with open-source software applications. However, the applications adopted include a mix of desktop (e.g., MyOffice), user interface (e.g., uPortal), and teaching and learning (e.g., Moodle) applications, all with adoption rates of 24% or less of the total survey sample.

*Summary of Findings on the Key Drivers of Adoption*

Figure 2 provides a graphical representation of the key drivers of open-source software adoption in U.S. higher education. At the top are higher education industry trends and pressures affecting institution stakeholders. To the left are the theoretical constructs from the literature. To the right are issues of interest to the academic side of the institution, and in the center is what technologists at institutions see as a win-win solution for all the institution’s stakeholders.

The research is fairly consistent in examining the key drivers of open-source adoption from the perspective of technologists. The overarching assumption is that
technologists are the end users and that adoption is driven by the needs and desires of those technical end users, be they philosophical, ease of development, role in the technology adoption life cycle, or perceived cost-effectiveness. However, faculty members and students using course management systems and institution staff members using administrative software applications are not necessarily technologists interested in what’s going on “under the hood” of a software system. Instead, their focus is on the ability to provide a learning environment grounded in the principles of constructivism and on achieving business efficiencies for institution staff members. The literature has not yet addressed the extent to which open-source software enables sound pedagogy, nor is there any empirical evidence of its effectiveness in bridging the gap between what technologists need and want and what academics need and want for teaching and scholarly work.

Future Directions

Although open-source software has been available to technologists for decades, it has only recently become available to nontechnical users. Open-source successes to date (e.g., Linux, Apache) and the more than 143,561 registered open-source projects in flight worldwide (SourceForge, 2007) primarily address the technical infrastructure of an organization’s data network systems or are programmer tools and utilities. There are only a relative handful of open-source products at the business application level, at which desktop applications such as Microsoft Office or campuswide administrative software applications such as student information systems, course management systems, and library systems reside. Moreover, the number of registered projects specifically intended for the education sector has declined somewhat from 773 in 2005 (SourceForge, 2005) to 663 as of March 2007 (SourceForge, 2007). Current education projects focus on a variety of areas, such as portals, classroom testing and assessment, library systems, and content development tools for computer-aided instruction. In 2006, the Andrew W. Mellon Foundation announced a grant for a group of research institutions to explore U.S. higher education interest in developing an open-source student information system (Wheeler, 2006).

International education appears to have been a lot quicker than U.S. education in researching and adopting open-source software, particularly for tools for the development of instructional content and delivery systems (Carmichael & Honour, 2002; Hirst, 2001; Leinonen et al., 2001). Consistent with European goals for free software development, deployment, and collaborative research, and the European Union’s desire to maintain its lead in the open-source arena (Ghosh, 2006), the University of Maastricht in the Netherlands and the University of Cambridge in the United Kingdom established a consortium focused on open-source projects, including the single largest knowledge base on open-source use and development worldwide. Similarly, a survey hosted by the University of Oxford (Cornelius, 2006) indicated that the use of open-source is on the rise at British institutions, with 77% of further and higher educational institutions considering open-source software in the software selection process and 25% mentioning open-source software in institutional policy. With respect to survey questions about virtual learning environments, open-source software dominates, with the Moodle course management system adoption at 39%, followed by Blackboard (19%) and WebCT (9%).
A variety of explanations have been offered for the differences between U.S. and international education’s deployment of open-source software applications. For example, in comparing the United States and Japan, Kerner (2006) contends that open-source adoption in the United States is driven from a developer perspective, whereas in Japan, open-source adoption is perceived as a strategic approach by system integrators and platform providers. Others point to the risks posed by the lack of engagement with the open-source community; past experiences with large, enterprise-wide software deployments; the threat of “forking,” whereby someone takes the software source in a different direction apart from the community; and the complexities of U.S. patent and trademark laws (Brooks, 2007; Wheeler, 2007) as reasons for slow adoption compared with international education.

To date, the three open-source software applications with high visibility in U.S. higher education are Sakai, an open-source course and content management system; Kuali, an open-source finance system for institution administration; and Moodle, an open-source course management system originally developed in Australia. Both Sakai and Kuali rely heavily on foundation grants to cover much of the development costs. In addition, nearly all Kuali and Sakai participants are large research institutions with relatively deep pockets and access to a broad pool of technical talent. However, doctoral and research institutions represent only 2% of all institutions of higher education in the United States. The other 98% consist of community colleges, to which research grants are few and far between, or small and medium-sized private and public institutions with limited resources, particularly in IT. Moreover, higher education researchers have not stepped up to the plate to conduct research about the advantages and disadvantages of open-source software in developing pedagogically sound learning environments.

Research needs to be conducted evaluating the long-term effect of these higher education–specific projects, particularly in terms of (a) the extent to which volunteer-developers address the quality and functionality issues of the broad base of community members, not just their home institutions or the community founding members; (b) what mechanism(s) these communities use for collaboration between their academic and technical constituents, to ensure that software applications are both technically efficient and supportive of engaged learning; and (c) how the learning outcomes and administrative efficiencies achieved with these open-source software applications compare with those of commercial vendor systems.

There is also an urgent need for research into the documentation and dissemination of proven models of the cost of ownership. With resource limitations a given, it is important for institutions to know what the true cost of open-source solutions will be, so that they can make informed decisions as to where they will invest their resources. To better understand the total cost of ownership of open-source software requires institutions that have already fully deployed open-source software applications to document and publicize the level of effort spent on open-source deployment and maintenance at their institutions. The number of IT staff members supporting each application, the skill sets of those staff members, their salaries, and the number of hours spent on open-source and basic institutional operations (hardware maintenance, etc.) would all have to be reported. More important, these metrics and measurements would have to be comparable across institutions. This would enable institutions to get a more complete and consistent picture of the
Adopting Open-Source Software

budgetary impact of open-source software and whether the savings in software license fees are true savings. As Villano (2006) noted, with only the top 200 U.S. institutions possessing the human resources necessary to implement open-source software and possessing the skills necessary to manipulate source code, institutions that have deployed open-source software applications have had to hire specialists or increase their training budgets for existing programmers to build and maintain open-source software applications. The result may be no savings at all and could mean increased costs. The bottom line: The absence of hard numbers will perpetuate what Green (2005) termed “affirmative ambivalence” toward open-source software, namely, agreement with the underlying concept but reluctance to actually adopt and to replace vendor product with open-source solutions.

Additional landmines on the open-source landscape concern regulatory requirements. For higher education, assessing the legal implications of using open-source software may be just as daunting as assessing its total cost of ownership. Higher education–specific regulations governing financial aid, the privacy of student records (Family Educational Rights and Privacy Act), accessibility (Section 508 and Americans With Disabilities Act compliance), and the privacy and financial regulations with which all organizations must comply (e.g., the Health Insurance Portability and Accountability Act, Sarbanes-Oxley Financial and Accounting Disclosure Information) are guaranteed in vendor maintenance agreements when institutions license proprietary business application software products. No such guarantees exist for open-source software. Researchers need to identify what options are available to open-source communities (e.g., support contracts covering regulatory compliance purchased from commercial software firms or talent in the area of regulatory oversight acquired and brought in-house) to address regulatory compliance and what impact those options would have on the total cost of ownership of open-source software.

Finally, researchers need to direct their attention to the challenges of intellectual property law and patents. Lerner and Tirole (2004) noted that to date, there has not been much systematic analysis of the implications of patents for open-source software. Moreover, a lawsuit filed in August 2006 by Blackboard, Inc., against e-learning software rival Desire2Learn for patent infringement (Carnevale, 2006) highlights the urgency of this challenge. Although Blackboard has pledged publicly not to challenge the Sakai and Moodle communities (Carnevale, 2007), the outcome of the lawsuit as well as other intellectual property questions are unknowns that education researchers must begin to tackle.

Conclusions

There is some indication that institutions of higher education in the United States view open-source software as the key to balancing pedagogical needs with the need for administrative efficiencies. In this review, I have outlined the five themes that dominate the software engineering literature and the education literature on the drivers of open-source adoption: (a) social and philosophical benefits, (b) software development methodology benefits, (c) security and risk management benefits, (d) software adoption life cycle benefits, and (e) total cost of ownership benefits. I have identified gaps in the literature in terms of evaluating open-source software as the key to capitalizing on ICT convergence to serve both teaching and learning needs and administrative needs. The literature is unanimous in deeming
open-source software to be for technologists; the overarching assumption is that technologists are the end users. However, this is not the case for open-source administrative applications or for applications intended to support teaching and learning. The literature provides little insight into how open-source software enables sound pedagogy or enables institutions to achieve a balance of sound pedagogy and technical efficiency.

A good starting point for future research is an in-depth examination of the progress and lessons learned from the Sakai, Kuali, and Moodle communities. Development, testing, and consistent, comparable documentation of models of the total cost of ownership will help dispel any myths associated with the real and hidden costs of open-source versus commercial software. Regulatory compliance and intellectual property law are also areas that require the attention of education researchers. Once a sound body of evidence begins to evolve, U.S. institutions of higher education will be able to make data-informed decisions that enable them to map their own resources, policies, procedures, and budget constraints with the “knowns” associated with open-source software adoption and the construction of integrated learning environments that achieve the desired academic and administrative outcomes.

References


Adopting Open-Source Software


Adopting Open-Source Software


Author

SHAHRON WILLIAMS VAN ROOIJ is an assistant professor in the instructional technology program of the College of Education and Human Development at George Mason University, 4400 University Drive, MS 5D6, Fairfax, VA 22030; e-mail: swilliae@ gmu. edu. She teaches courses in the e-learning graduate certificate program using course formats that model the various e-learning delivery modes and emphasizing the application of instructional design principles and learning theories to e-learning in both educational and training settings.