

By Brad Wheeler

OPEN SOURCE 2007

Developing sustainable economics and advancing the frontiers of innovation are the dual challenges for application software in higher education. *Sustainable economics* means that an institution's base budgets can support the licensing fees, developers, maintenance, training, and support required for application software. For example, it means that the viability of a course management system (CMS) is not dependent on the next grant or on a one-time

How Did *This* Happen?

budgetary accommodation. Since making changes to application software invokes cost, minimizing change is one tactic for achieving sustainable economics through lower IT costs. In higher education, however, the creative nature of colleges and universities motivates faculty and staff to innovate with new pedagogy and with the use of online resources. Application software that fails to evolve or to allow experimentation and innovation in teaching is unlikely to be well received.

Brad Wheeler is an associate vice president and dean of IT at Indiana University. He is also an associate professor of information systems in IU's Kelley School of Business.



Higher education is in search of a new model to address these dual challenges, and open source application development has been proffered as a solution. Open source software, which is usually obtained without paying any licensing fee to its creators, allows developers to modify the inner workings of the software. In contrast, commercial application software, which is usually licensed for an annual fee, does not allow the inner workings of the software to be modified. Open source software is not free, however, when properly viewed from a total cost of ownership (TCO) perspective. Like all other systems, it requires investments for hardware, user support staff, training, integration with other systems, and so forth. Thus licensing fees, technical support, and control of destiny in evolving the software features are the discriminating cost factors. But licensing fees are not trivial: some estimates place licensing costs at 20–25 percent of the TCO—in the hundreds of thousands of dollars for many institutions.¹

Software developed using the open source model has demonstrated remark-

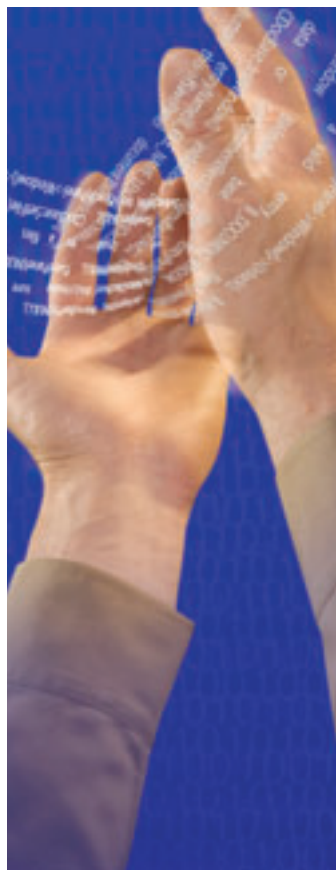
able success for widely used operating systems and infrastructure software. This type of software resides between the physical hardware and the application software, and its capabilities are used in many industries. Linux, Apache, and JBoss have garnered enormous individual and corporate followings, with a number of companies providing for-fee commercial support without charging a license fee for the free software itself.

Can open source success at the infrastructure software level translate into similar success for application software? In 2002, at the dawn of open source applications for higher education, Thomas Warger asserted: “Very little is known about how this [open source] mode of collaboration would work. The academic community is an ideal place for such an experiment—given the dispersion of talent among so many institutions—but it is also a difficult environment for the experiment because of the strong tradition of local independence.”² Similarly, Gartner asserted that higher education would likely be the first place for porting the open source model to application

of open source application software projects listed in Table 1. Each project represents some blend of partnering, institutional investment, foundation/government investment (Mellon, Hewlett, NSF, etc.), and community building. Some institutions have publicly proffered open source applications as a major part of their strategy, whereas others point to a number of important questions that are not yet answered.⁴ In 2004, the open source model can offer only a hopeful path to sustainable economics and innovation. It remains unproven for large-scale application software.

Scenarios

This section offers a look at two possible outcomes from porting the open source model to application software in higher education. The time is set as 2007, three years hence, and the scenarios are written as news articles. In the first outcome, the open source model has had a broad and enduring effect on the industry. In the second outcome, the model has had a negligible or only modest effect. Success is defined in terms of how well the open



Can open source success at the infrastructure software level translate into similar success for application software?

software, with academic applications emerging first and administrative applications to follow (if the academic applications were successful).³

The increased interest in open source is well timed as higher education seeks a new model and as IT infrastructure becomes more homogenized. TCP/IP for networks and Web browsers on every desktop are ubiquitous standards. Many colleges and universities have chosen Linux, Java, and Apache as IT standards. Data specifications from IMS and others have provided data-exchange interoperability among systems. This homogeneity in skills and IT infrastructure is an antecedent for a new model based on collective, inter-institutional investment.

Nascent confidence in this new open source model is illustrated by the number

source model addressed the dual challenges of developing sustainable economics and advancing the frontiers of innovation as indicated through adoption of open source application software.

Scenario 1: Open Source Becomes Mainstream

(September 17, 2007) What began as a noble experiment in inter-institutional cooperation has now emerged as a viable “community source” model for mainstream application software. Only a few years ago, colleges and universities had only two options for application software. Some chose to build their own systems, using large IT staffs or contractors, while most chose to license commercial products from the few leading vendors in each category of software. The community

Table 1. Some Open Source Application Projects in Higher Education

SOFTWARE PROJECT	INSTITUTION(S)	URL
Chandler/Westwood	Open Source Applications Foundation; 30 universities	http://www.osafoundation.org/
DSpace	MIT Libraries; Hewlett-Packard (HP)	http://www.dspace.org/
Fedora Project	Cornell; Virginia	http://www.fedora.info
LionShare	Penn State	http://lionshare.its.psu.edu/main/
Open Knowledge Initiative (O.K.I.)	MIT	http://web.mit.edu/oki/
Open Source Portfolio Initiative (OSPI)	Carnegie Foundation for the Advancement of Teaching; Delaware; Indiana; Michigan; Minnesota; Rhode Island; the r-smart group	http://www.theospi.org/
Pubcookie	Washington	http://www.pubcookie.org/
Public Key Infrastructure (PKI)	Dartmouth	http://www.dartmouth.edu/~pkilab/
Sakai Project	Indiana; Michigan; MIT; Stanford	http://www.sakaiproject.org/
uPortal	JA-SIG member institutions	http://www.uportal.org/
Visual Understanding Environment (VUE)	Tufts	http://vue.tccs.tufts.edu/
WeBWorK	Indiana; Rochester	http://webwork.math.rochester.edu

source model, based on open source principles, promised a yet-to-be-proven third option. A recent survey has shown that over 72 percent of colleges and universities have already deployed or have committed to deploying open source applications for course management, library, or other academic applications. The recent initial releases of administrative software show similar promise.

Quality Applications. The Sakai Project provides one example. In 2004, a joint investment by four universities and a grant created the Sakai Project. Its open source Collaboration and Learning Environment (CLE) software quickly evolved into a mature and feature-rich application. The software is now in use at small liberal

arts colleges, community colleges, large universities, and quite broadly outside of the United States. Although its core features—service delivery through uPortal, calendaring, threaded discussion—are arguably undifferentiated from the features of various commercial offerings, the extraordinary explosion of discipline-specific learning tools has been impressive. “We’ve seen leading faculty and disciplines develop and share many very clever tools to help students learn specific domain content,” said one member of the Sakai Educational Partners’ Program (SEPP). Faculty in mathematics, engineering, languages, and other disciplines have made their teaching tools open source and compatible with Sakai software. The partners and others have

propelled a rapid adoption and refinement of these modules and of the Sakai core while giving their enhancements back to the community.

The Open Source Portfolio Initiative (OSPI) software, now in release 4.3, provides a second community source example. It has become the most widely used portfolio application. Its user community has developed a number of tools that have helped institutions begin to realize the benefits of portfolio-based reflection and assessment. All major CMS vendors have import/export options for or direct interface connectivity with OSPI software. Many administrators have been surprised at how quickly a number of disparate open source projects have become interoperable, which they view as a boon to reducing implementation and support costs. “The fact that our library systems use Fedora, OSPI uses Fedora, VUE uses Fedora, OSPI uses VUE, OSPI integrates with Sakai and uses PKI, and they all use uPortal and OKI has far exceeded my expectations for integration,” said one CIO. “It has really changed our approach to procuring applications. We can now see an application working at another institution and bring it here with modest implementation effort. Amazing.”

Reliability and scalability were also viewed as threats to open source viability. “Not all open source code is of the same quality,” noted another CIO. “You must maintain some perspective. We did have some initial performance issues with one of our applications, and we quickly realized that our concerns were actually community concerns. We were able to jointly iron out the performance issues and share the improved code for the next release.” It is not surprising that some of the large community college systems and state schools have the biggest stake in solid scalability for hundreds of thousands of users. “If it reliably scales for the big schools, then I know that the performance is there for a place our size,” said one IT manager.

Successful Joint Ventures. With few exceptions, the most broadly adopted open source applications thus far are the products of joint ventures that create communities. The pattern is that two to four institutions pool their resources—



vendors' products to fill software needs, and we continued to do so while their products worked and were economically viable. We'll continue to engage in these open source communities under the same criteria." The early uPortal project demonstrated the value of communities—including vendors—to sustaining the software. For-fee programs, like the Sakai Educational Partners' Program or JSTOR, have focused on

universities move to a standard license model for their open source contributions. "Sadly, it's been UNIX redux," said one developer familiar with the situation. "It will take several years, if ever, to reconcile the existing code to a license structure in which we can share code without invoking university counsel. This was so preventable a few years ago when we were just starting."

Beyond the features of the software and the means of its creation, sustainability is the number-one issue.

often developers and other staff—to develop software for a specific need. Foundations have been especially active in providing matching funds to accelerate the coordinated development. "The market is speaking," said the program director of one foundation. "Our grants are leveraging the real investment that universities put on the table. When universities vote with their pocketbooks, we know there is a real need for the software and for our financial help to make it so." Some institutional administrators wonder if the foundations will continue to support development or if they will turn their attention to something else. Others point to community source successes that have been accomplished solely by pooling institutional investment, sometimes with corporate support but without foundation dollars.

The Community. Beyond the features of the software and the means of its creation, sustainability is the number-one issue. "When you buy a vended product from a company, you do your due diligence, escrow the source code, and negotiate hedges for potential risks," said one veteran CIO. "With open source software, how do I know that the community cooperation that works today won't dissipate next year? The Apache Web server has the Apache Software Foundation to ensure its evolution. What is the equivalent in higher education?"

"It is the economics," responded another CIO. "We've historically purchased

community building to replicate those early experiences.

Licensing. One troubling spot for open source applications is the ongoing mish-mash of inconsistent licenses. Although all projects assert that they are open source, code that originates at one institution may still carry minor restrictions that impede its bundling with other code. Some early licensing concerns and differing philosophies have caused friction among open source advocates in higher education. Various copyright, copyleft, and viral restrictions imposed by institutional technology-transfer offices have caused some very good open source tools to remain outside the community.

For example, one open source project acknowledges a very good add-on tool that was created at another institution, but it refuses to add the tool to the official releases or to post the add-on to the project's CVS (Concurrent Versions System) tree because of the differing license terms. "We have to protect the license lineage of source code for this application," said the project leader. Institutions that want to adopt other tools under other licensing arrangements are certainly free to do so directly from the tools' creators.

There is some hope for a resolution to the licensing challenges. EDUCAUSE recently published two template licenses that serve differing licensing objectives. This has helped a number of colleges and

Evolving Support. The market for support services provided by commercial Open Source Support Providers (OSSPs) appears to be entering an industry shake-out. The strong flow of new software and tools around a few core applications has allowed some OSSPs to create support suites across a range of applications at favorable pricing. "We just added commercial support for PKI and DSpace as part of a multi-application, multi-campus support agreement from a leading vendor," said a community college vice provost. "The pricing was very good, and we contracted for DSpace integration work as part of a new institutional repository initiative." Not all vendors are choosing to support a range of open source applications, and some applications continue to lack support options. Commercial support for some smaller applications remains a practical monopoly, and two firms have recently exited the OSSP market, saying that it was unprofitable.

Why Not Both? Another surprise for open source applications has been the extent of interoperability with commercially licensed software. "We've found the ability to stick with our commercial CMS while blending it with the broad innovation in add-on, open source tools to be very effective," said one IT leader. "We have a strategy that uses commercial applications for our core systems for CMS, library, etc., but we've insisted upon open

standards that work with open source software as a contracting condition.” Real open standards for interoperability have matured considerably through reference implementations from the open source community. This has helped everyone overcome vendor rivalries when implementing presumably agreed-upon specifications.

Scenario 2: Few Open Source Applications Matter

(September 17, 2007) What began as a noble experiment in inter-institutional cooperation for developing open source application software remains without broad adoption in higher education. “I won’t yet say that open source is dead as a model for higher ed, but you hear a lot less talk about grand open source or community source visions these days,” said one university CIO. “We continue to track it and have found some nice tools for certain niches, but the big projects of a few years ago remain far less than we had hoped for.”

What happened? The open source model promised a third way, an alternative to the age-old choice between building/supporting custom software and risking vendor lock-in with commercial applications. Institutions large and small signed on and invested cash and staff in collaborative efforts, yet almost none of the resulting open source applications can claim broad adoption or idea leadership. Interviews with members of these open source projects and adopters of the software point to one recurring theme: “We just couldn’t agree.”

among competing departments and schools at a single university. Those are multiplied when you try to set a direction based on money and requirements gathered from many institutions.”

The lack of agreement is rooted in more than just political challenges. Each college or university evolves at its own rate in adopting new technology architectures, requiring new functionality, and implementing changes. The open source vision of easy software sharing among institutions has not yet been able to overcome these institutional timing differences. Open source projects for libraries or CMS have not even been able to agree on common technologies and standards among themselves. “The systems integration work of implementing several open source applications is just as bad if not worse than integrating commercial packages,” noted one industry consultant. “The savings from not paying commercial licensing fees are quickly absorbed by the substantial implementation challenges of open source. The university technologists like the ability to tweak the code, but frankly, the economics just aren’t there.”

Inability to Leverage. The concept of pooling institutional staff talent and investment toward shared software had obvious appeal as open source projects gained momentum a few years ago. “They can agree to pool their money, but they have not learned how to dance together,” said one industry observer. “It’s like watching a bad tango dance performed by a couple with four left feet.”

to go “local” during times of project conflict proved overwhelming.

Unlike other open source projects, such as Linux or Apache, that leveraged extensive networks of individual volunteer programmers, most higher ed applications have been designed as community source projects for contributions from institutional developers. Most project leaders can point to a few quality contributions from individuals, but all acknowledged that such contributions are rare: “We don’t know how to leverage the solo programmer or independent consultant out there who just wants to contribute something of interest. It doesn’t fit our model, and we cannot depend on this group to do things that hit release dates.”

Even when various open source projects do create good code, licensing differences hinder the ability to leverage the work across projects. Colleges and universities have a legitimate interest in setting the licensing terms for the work products of their development staff. The patchwork of licenses, with minor discrepancies in their rules for use, derivative products, and commercial restrictions, has created an enormous drag on the exchange of reusable software modules. Attempts by EDUCAUSE and others to propose template licenses for higher education have had little effect. Some university technology-transfer offices still frown heavily on open source licensing of university-created software that might have potential commercial value.

Perhaps Not Yet. The shared development model is working for some institutions as

Each college or university evolves at its own rate in adopting new technology architectures, requiring new functionality, and implementing changes.

A Failure to Agree. “If I were the CEO of a company, then I could say: ‘We are using this architecture, the product will ship on this date, and the feature set will include A, B, and C,’ ” explained the project leader of one open source endeavor. “You can’t do that in inter-institutional collaboration. Think of all the challenges

The distributed staff resources—such as developers, usability specialists, and database specialists—offered a treasure of seasoned talent to the open source projects, but most never learned how to coordinate in geographically distributed development. Thus, the coordination costs, the misunderstandings, and the instincts



they collaborate in small groups. An administrator at one research university commented, “We get tremendous value from our core partners in sharing maintenance work on the software.” That statement suggests some value in small, core circles, but it does not indicate that open source would be a means toward leverage in the industry. Perhaps it is simply too early to judge if the open source “noble experiment” will pay off for higher education.

Which Scenario and Why?

Which of these scenarios is more likely to foretell the headlines of 2007? Will the outcome be determined by the structural attributes of the higher education industry, or can an outcome be chosen through collective action? An analysis of these two scenarios reveals three important considerations for understanding the future of open source. First, the task of developing application software is very complex and

variety of computing hardware and architectures. And it is complex in that users invoke its features in countless sequences and unintended ways.

Thus, creating and sustaining reliable and scalable software requires considerable resources for planning, coordination, and precise execution. These resources are expensive, consuming millions of dollars of investment over the life cycle of the code. Resources to engage the task must be aggregated and then effectively deployed over a sustained period of time. It is this very resource requirement over time that causes many institutions to rethink the wisdom of home-grown systems.

Coordinating Mechanisms

Resource aggregation has typically been accomplished through two coordinating mechanisms: an *aggregation of capital* or an *aggregation of will*. In 2004, commercial organizations are the dominant developers

Both mechanisms—the aggregation of capital and the aggregation of will—recognize that the challenges of creating complex software differ from the challenges of sustaining it as users’ needs diverge and IT architectures inevitably evolve. Creating the software is easy relative to sustaining it. Mergers, acquisitions, and new business strategies sometimes threaten the continuity of commitment from commercial organizations over the years, whereas a weakening of will, bad politics, and lean budget years can threaten continuity of commitment in consortia.

If the community source model proves viable, it will do so because it is an economically efficient coordinating mechanism for software investments in higher education as an industry. An analysis of any historical software system—online card catalogs, Web-based registration, course management systems—over a five-year period will reveal



Creating and sustaining reliable and scalable software requires considerable resources for planning, coordination, and precise execution.

expensive, and sustainable models must address this complexity. Second, coordinating mechanisms will arbitrate success. And third, the collective actions of institutions will influence the outcome.

Complex Software

Modern application software is complex—especially software that meets the diverse needs of many institutions. It is complex in the number of elements (lines of code, number of modules), in the many-to-many interactions among those modules, and in the precise timing under which those interactions must succeed while serving thousands or even one hundred thousand users. The software is complex in that it must work reliably on a

of mainstream software. They use an aggregation of capital. They set aside upfront working capital, discern a market opportunity, hire staff, develop the software, and license the software to a large number of institutions, along with gaining potential profits from maintenance or consulting. Profits from this activity enable further resource aggregation to sustain and evolve the software.

The community source model gathers the resources to create complex software through an aggregation of will. A few adopting institutions essentially “pre-buy” the software by contributing staff or cash to the development effort. They gain some direct input into the software design and express their will through an inter-institutional consortium, board, or project office. The aggregation-of-will coordinating mechanism may be supplemented by one-time investments from foundations that support higher education.

that individual institutions separately invested hundreds of millions of dollars in home-grown or commercial software. Can that flow of higher education resources be harnessed to create better economics and shared innovation outcomes for everyone?

Apache and Linux exist as successes today because both projects found efficient models to aggregate the needed resources—mostly through the voluntary contributions of individuals—to overcome the complexity challenge of infrastructure software. They were developed in the pure sense of the open source software movement, and they provided coordination and synchronization of development efforts.

The community source model for application software in higher education has some similarities to, but many important differences from, those projects. It is similar in that community source is

largely about synchronizing the investments of contributors. The resulting software is freely available for use or modification, with the anticipation that the free software will encourage others to contribute software enhancements to the communities. But the community source model is different from those infrastructure projects in that most higher education applications are too small to attract and retain an army of individual volunteer programmers (as with Linux). The contributed resources are institutional staff time and cash for particular projects. The evolution of the authoritative release



aggregation-of-will approach through a community source model is only now being tested as a coordinating mechanism for industry software investment.

Collective Actions

Community source development models will succeed or fail based on collective decisions made among institutions. Thus, we can identify institutional behaviors and actions that will favor or that will likely fragment community source efforts.

Actions Favoring Community Source Coordination. Community source development will prosper when institutions can use collective projects to synchronize their institutional clocks for software applications. For example, a project in which four institutions collectively develop electronic portfolio software in one year is likely to yield better economics and functionality than a project in which four institutions individually create disparate portfolio software over several years. Thus, the first challenge for participants in a community source project is to find like-minded partners who share a similar vision and timeline.

Beyond synchronizing software initiatives, colleges and universities must also

strategy will not likely develop effective partnering skills and will confront very high systems integration costs from a series of uncoordinated choices.

Actions Fragmenting Community Source Coordination. Warger asked if higher education institutions can overcome their “strong tradition of local independence,” a tradition that has neutered collective leverage. The historical answer has been no: commercial coordination for software development remains the incumbent model. Colleges and universities pay licensing and maintenance fees, and companies coordinate the development and support for a profit. In the absence of directed changes in institutional software investment, the commercial model will continue without an alternative.

Four institutional behaviors threaten community source success. First, colleges and universities might choose to wait out the uncertainty, not engage in community projects, and hope that community source projects succeed without their participation. This behavior robs community source projects of the resources they need to create open source software. Second, institutions that want community source to create a third option might be lured by

Community source development models will succeed or fail based on collective decisions made among institutions.

of the software is governed through formal boards and representative mechanisms of the institutions.

Thus, projecting all of the good or ill of the open source movement onto today's community source projects for applications is likely to yield little real insight. Higher education is in largely new territory here, and this territory is perhaps uniquely accessible to our industry. Few competitive industries have a culture or incentive to collaborate in shared software development. For example, it is difficult to imagine Ford and General Motors working together on an open source billing system. Higher education's

develop a competency to partner well. If partnering for software development were easy, it would be the norm today. It is not easy. It requires a cooperative mindset, disciplined choices among staff, and leaders' consistent vision that the value of partnering over the long term exceeds the easy short-term gains of defecting to local priorities.

Finally, participation in community source efforts is not for those who want to dabble. Any value from community source, synchronized development will likely emerge through a disciplined and consistent IT strategy sustained over the years. Organizations that fail to choose a

short-term commercial deals that are too good to refuse. Successful community source software is likely to put downward price pressure on commercial licensing, and incumbents may invoke unusually low pricing approaches in the short term to dissuade institutional investments in community source projects. Third, colleges and universities might not invest sufficient resources in local open source systems implementations and then blame the software for a failed project. Companies have faced this undue recrimination many times, but the effect could be especially chilling on the cooperative nature of community source collaborations.

Finally, the greatest risk is that higher education institutions simply might not cooperate. Ineffective governance of projects, unproductive debates over technology nuances, or failure to sustain an IT strategy over multiple years could all impede community source success.

Guidance for Stakeholders

How can stakeholders establish a strategy to advance their goals for community source development in these early and uncertain years? This section provides guidance for higher education, for foundations, and for commercial stakeholders.

Higher Education

Sustainable economics and innovation in information technology are more likely to be realized when IT investments are guided by an IT strategy. An IT strategy should provide guidance for technology standards, staffing levels, and roles for vendors or partners. It should establish a philosophy for choosing what to do in

If a decision is made to participate in a project and to use community source software, implementing the decision requires clear communication and consistency among staff. For example, developers need to understand that the institution's strategy is to leverage the work of others and that projects veering from that strategy will require strong justification. Most important, CIOs, provosts, and presidents must understand that developing effective community source coordination mechanisms through an aggregation of will necessitates strong inter-institutional leadership over multiple years. This is not a short-term fix for current budget challenges; rather, it is an investment in improving the long-term economics of information technology for an institution and for higher education.

Like most other enterprises, colleges and universities hope to engage in gainful trade, and they do so under fiduciary responsibilities to their stakeholders. They pay cash for software licenses, or

particular tool or extension to the software. These examples show how institutions can spend their currency—either cash or tendered staff time—to trade for the software they need through leveraged community source projects.

Foundations

Foundations, both large and small, continue to be an essential part of advancing higher education, and most operate with particular objectives for grants. One common objective is to help colleges and universities use their own resources in ways that most directly benefit student learning and research.

The vision for community source projects—to improve the economics of and innovation in software for higher education—is a natural fit for the objectives of these foundations. Foundation boards and program officers are encouraged to use their considerable influence to fund innovation but to do so in ways that create sustainable

Software developed using the open source model has demonstrated remarkable success for widely used operating systems and infrastructure software.

order to avoid fragmented, tactical IT choices that are unlikely to serve well over the years.

College and university leaders should deliberately assess how the success (or the demise) of community source efforts could affect their objectives. If the availability of open source software without annual licensing fees is desirable, is the institution ready to financially participate in and make use of the community source projects? For example, if a number of community source projects are developing for Java/Linux platforms, does the institution have the skills and technical architecture to use this software? Is the institution effectively monitoring relevant community source projects and tracking those that are of interest? If the use of community source would entail a shift in IT strategy, have the appropriate administrators and faculty been engaged in the discussions?

they hire staff to develop local systems. Community source projects provide another form of exchange. For example, each university in the Sakai Project is investing \$1 million in dedicated staff time. The aggregation of these in-kind investments plus foundation support means that \$1 million investment leverages \$6.8 million in project work. Other institutions can spend nothing and download the full, free software from the Sakai Project, but they will not have had any influence in the evolution of the software, nor will they have gained the considerable skills from inter-institutional knowledge-sharing.

Other institutions can invest \$10,000 annually to get the services of the Sakai Educational Partners' Program, which provides technical insights, staff, and training. SEPP members can also choose to invest a developer or cash in collaboration with other SEPP members to build a



economics through interoperability with other community source projects.

For example, the University of California at Merced is a startup campus. It has few legacy systems and could set an IT strategy to fill many of its needs through open source software. Think how much could be saved if new foundation grants begin requiring interoperability with a set of core community source projects. These conditions, with the grants, could pay back hundreds of times across the industry as they reduce the costs of adoption and integration for all.

Commercial Stakeholders

The software created by the commercial model will continue to thrive alongside community source software if companies can create bundles of software and services that provide compelling value for colleges and universities. The future, however, does represent a discontinuity in the software market for higher education, and wise company leaders will adapt

to that reality rather than hoping to prosper while ignoring it.

The first discontinuity is in the value assigned to proprietary intellectual property (IP) via the pricing mechanisms of licensing agreements. Both vendors and customers alike contend that the current licensing model of upfront payments and then fee-for-upgrade is not serving either group very well.⁵ Subscription-based licensing models tied more closely to actual use appear likely, but what value will institutions assign to proprietary IP licenses when (arguably) comparable software is available without licensing fees? Vendors must carefully consider if their considerable investments in creating and sustaining proprietary IP will profitably cover their costs.

A second discontinuity for vendors is the double-edged sword of creating true interoperability while selling unique value. Customers want open systems using open standards, but the very act of truly achieving this commoditizes the

proprietary value of a system's uniqueness. Community source projects also create viable reference implementations of open standards that favor no particular company. With these reference implementations, customers who want to mix and match commercial and community source software will have an unprecedented tool to enforce real interoperability beyond contradictory claims of supporting open standards.

Finally, commercial interests must carefully consider their choice of strategy and the requisite organizational capabilities. One strategy is to continue focusing on selling bundles of proprietary software and support in a manner that demonstrates the value of the commercial coordination of software. A different strategy is to partner with community source projects for open-license software creation and then compete on for-fee consulting, support, and integration services.⁶ The latter strategy will require new capabilities in partnering with community source

projects for leveraged development, in creating a less costly selling process, and in developing a profitable new business model that is not based on controlling the IP. A blended strategy that combines open source and licensed IP may be another path. Over time, customers will choose from the community source and commercial worlds those offerings that best meet their needs. Savvy business strategists know that times of industry discontinuities provide companies with the greatest opportunities (and risks).

Conclusion

The network era has imposed many new software costs (and benefits) on higher education. Colleges and universities are in search of a new model to fund application software as they struggle to “do more with less.” In 2004, Warger’s anticipated experiment in academic community collaboration is already well engaged. Community source projects—based on open source philosophy and licensing—prom-


ise sustainable economics and innovation through coordinated industry investment. The outcome of this experiment—the success or failure of the new model—is not predetermined. Rather, it will be the by-product of individual and collective institutional actions. *e*

Notes

1. Michael Arnone, “Course-Management Outfits Still Seek Elusive Profits,” *Chronicle of Higher Education*, July 12, 2002; Jeffrey R. Young, “Pricing Shifts by Blackboard and WebCT Cost Some Colleges Much More,” *Chronicle of Higher Education*, April 19, 2002; Tom Head et al., “Setting a Next-Generation CMS Strategy” (presentation, EDUCAUSE Annual Conference, Anaheim, Calif., November 7, 2003).
2. Thomas Warger, “The Open Source Movement,” *EQ: EDUCAUSE Quarterly*, vol. 25, no. 3 (2002): 20, originally published in *Edutech Report*, vol. 18, no. 2 (May 2002).
3. Ron Yanosky, Marti Harris, and Michael Zastrocky, “Higher Education E-Learning Meets Open Source,” *Gartner*, December 16, 2003.
4. Annie Stunden, “The Muscles, Aches, and Pains of Open Source,” *EDUCAUSE Review*, vol. 38, no. 6 (November/December 2003): 100–101, <<http://www.educause.edu/pub/er/erm03/erm03611.asp>> (accessed May 3, 2004); Howard Strauss, “The FREE, 0% APR, Better Sex, No Effort Diet,” *Syllabus*, November 1, 2003, <<http://www.syllabus.com/article.asp?id=8460>> (accessed May 3, 2004).

5. Gregg Keizer, “Software Licensing Said to Be Set for Dramatic Shifts,” *InformationWeek*, March 23, 2004, <<http://www.informationweek.com/story/showArticle.jhtml?articleID=18401482>> (accessed May 3, 2004).
6. Brad Wheeler, “The Inevitable Unbundling of Software and Support,” *Syllabus*, March 1, 2004, <<http://www.syllabus.com/article.asp?id=9026>> (accessed May 3, 2004).

RELATED RESOURCE

 The ECAR publication “Aligning IT Strategy to Open Source, Partnering, and Web Services,” by Bradley C. Wheeler, *EDUCAUSE Center for Applied Research Bulletin*, vol. 2003, no. 24 (November 2003) (<http://www.educause.edu/asp/doclib/abstract.asp?ID=ERB0324>), examines the current state of open source initiatives in higher education and provides a roadmap for incorporating open source application development into an information technology strategy.