

EDUCAUSE Center for Applied Research

Research Bulletin

Volume 2007, Issue 1

January 2, 2007

# Considering Open Source: A Framework for Evaluating Software in the New Economy

Lois Brooks, Stanford University



## Overview

Major investments from universities, along with seed funding from foundations, are producing a host of software applications for use in higher education administration, learning, and scholarship support. With the promise of new choices for enterprise applications, a higher education–focused approach, and greater control of information technology (IT) environments, the software is sparking interest in potential adoption from institutions worldwide. Simultaneously, the community source movement is raising questions for administrators about when and whether to adopt or devote resources to software development projects—provoking questions of sustainability, future directions, and total cost of ownership. This research bulletin frames the issues an institution should consider with respect to adding community source products to the portfolio of software, infrastructure, and services that constitute the IT environment.

## Definitions

In the evolving worlds of open source and community source software, we consulted Wikipedia, itself a community-edited encyclopedia, for current definitions of these terms. According to August 5, 2006, definitions:

- **Open source** describes practices in production and development that promote access to the end product’s sources.... Open-source software is computer software whose source code is available under a copyright license that permits users to study, change, and improve the software, and to redistribute it in modified or unmodified form ([http://en.wikipedia.org/wiki/Open\\_source](http://en.wikipedia.org/wiki/Open_source)).
- **Community source** builds upon the open source production and development models but introduces more formal commitments and coordination to the work and strategic directions. In community source, efforts are explicitly coupled to defined roles, responsibilities, and funded commitments by community members rather than the often voluntary commitments made in pure open source. Importantly, the community source model will incorporate commitment from individuals that are sanctioned and supported by their host organizations, e.g., their employers. This also manifests itself in the development of the community source development reflecting the strategic needs of the host organization. All of the results of the community source model must be open source in licensing terms, but control of the code base or content base is retained by the coordinating community ([http://en.wikipedia.org/wiki/Community\\_source](http://en.wikipedia.org/wiki/Community_source)).

## Highlights of Open Source Considerations

For decades, higher education has wrestled with costs and benefits of buying or building software. The transition from homegrown software to commercial software became possible when institutions could take advantage of the economies of scale offered by software vendors who offered reasonably priced, somewhat flexible enterprise solutions

that served the needs of most higher education institutions. While some institutions have been well served by commercial software, others have been disappointed. In many instances, costs have skyrocketed because vendors charge for custom adaptations. In other cases, commercial products were shoehorned into an academic environment, providing less functionality than promised, requiring expensive local customizations to meet academic needs, and locking institutions into single source contracts. When customizations were extensive, upgrade costs became prohibitive, and institutions were caught in the worst of both worlds: locked into software that is expensive to upgrade and to maintain. Even without institutional customizations, costs have grown so high that many institutions have never been able to reap the benefits of commercial software (R. Livingston, personal interview, April 7, 2006).

It is no surprise, then, that some institutions opt to develop their own software, but for enterprise-wide applications the costs of doing so can be prohibitive. As a result, the initiatives in community-based development are a natural evolution. Pooling resources to develop software for multiple institutions offers participants a way to achieve local goals while spreading costs more widely. Groups of universities, commercial providers, and in some cases, consortia, are undertaking projects to develop software together. The idea is simple: codevelop functional specifications and project goals, then pool programming resources to have one product for everyone. Products are released as open source, meaning that the software code is freely available for anyone to use and modify. While some projects have received a lot of publicity, they are not the only open source options available. Hundreds of products are available for adoption at all levels of the technology stack. Rob Abel summarizes the current environment succinctly:

The business drivers for open source applications in higher education favor traditional non-open source providers if they are responsive to buyers [sic] concerns. From the buyer side, the value proposition for open source applications can be summarized as combination of cost, control and the possibility for innovation. The uncertainty in the commercial market being able to provide the higher education specific products in emerging critical areas such as course management and portal in a stable, predictable and affordable manner trumps any specific gains in functionality being sought. (Abel, 2006)

## Community Source Trends

As community source projects are maturing, several trends can be observed:

- Some projects, such as Sakai and the Open Source Portfolio, are evolving as a suite, with interoperability and common practices emerging and communities overlapping. This is driven by the interests of people installing the products and supporting the software, whose portfolios are wider than just one product. Interoperability is a necessity to meet scholarly needs, so collaboration and cooperation among projects becomes a mandate.
- Some projects, including Fedora, Linux, MySQL, uPortal, DSpace, and Sakai, are reaching second and third generations of software. In thriving projects, early

decisions are being reconsidered and reprogrammed, and emerging standards and technologies are being adopted.

- After early waves of hype subside, projects such as Moodle, Fedora, and Sakai are developing core groups of contributors and vibrant user communities.

These trends are similar to those of the commercial software world, where an installed user base and iterated versions of software are indicators of stability. Institutions must consider stability in selecting commercial and open source products. In particular, coordinated evolution of software products, which will drive interoperability and harmonious usage on campuses, matters for the long-term viability of an institution's software choices. Bob O'Leary, Executive Director of Administrative Systems at Stanford University, said, "It's all about the long term. You're not just solving today's problem." A product's ability to maintain the interest and commitment of the academy, whether through commitment of fiscal resources (in the commercial software world) or human resources (in the community source world), are necessary for its long-term viability (R. O'Leary, personal communication, May 10, 2006).

Over time, the perception of both commercial and open source software has changed, as higher education administrators have gained expertise in enterprise applications. A study undertaken by the Alliance for Higher Education Competitiveness offers a snapshot of current perceptions of software in higher education. Strengths of open source are perceived to be the total cost of ownership, integration with campus infrastructure, and better functionality and security. Weaknesses are perceived to be need for more staff skills, limited options for commercial support, and product maturity. Overall, university administrators have a positive perception of the total cost of ownership of open source, with 56 percent believing that open source is less expensive in the long run, 37 percent expressing no perception of cost advantage, and only 7 percent perceiving that open source is more expensive to maintain than commercial applications (Abel, 2006). Given changing perceptions, an objective evaluation now considers products from commercial and community sources.

The buy-or-adopt decision requires analysis of acquisition, customization, and integration costs in the initial phases of systems deployment, upgrade costs and maintenance through the product life cycle, and exit strategies for the inevitable next generation of products. While some criteria of open source and commercial software choices are the same, there are some differences as well.

## When Panacea Meets Reality

A period of excitement and enchantment accompanies every opportunity to develop or buy new software—a sense that with a bit of money to invest all problems will go away. As projects progress, project teams learn that integration, deployment, and user acceptance are more difficult than expected, and often the outcome is only incrementally better than the software that the new product replaces. Partially, this is because software does not change underlying structures of business processes, architecture, and staff skills (R. O'Leary, personal communication, May 10, 2006). These environmental conditions are critical to the decision to adopt or purchase software. Because of the

inherent flexibility of open source software, decisions about how to use and integrate the software can be made during the deployment project and throughout the product life cycle, so environmental conditions may have more impact on the project's success than if using a turnkey solution. For example, as Stanford deploys Sakai, the decision was made to customize the way cross-listed courses are represented, requiring modifications and additions to the source code but better integrating with Stanford's complex course structures. This choice would not have been possible in a closed system. With each new release of the Sakai software, the Stanford team evaluates what will be deployed locally.

In considering whether to adopt and how to make use of open source software, decision makers should consider several variables:

- The architectural and integration goals of the institution
- The trade-off between flexibility and internal control versus a turnkey solution
- The trade-off between internal versus external support
- The existing and desired core skill sets of staff at the institution
- The stability and future of the companies and communities

## Open Source as Part of the Overall IT Portfolio

The goals of any enterprise system deployment are to develop a more cost-effective way to conduct a business practice and to minimize software costs in the long run. To meet these goals as well as the promise and challenge of managing enterprise systems in higher education, IT portfolios must be increasingly modular, complex, and interoperable. Business systems for finance, human resources, and student information management must work with authentication and authorization systems, the campus portal, the library system, course and content management, and so on. There is increasing demand for e-mail, calendars, and similar utilities to work seamlessly with enterprise systems. Imagine a scenario in which a change in a class room assignment automatically updated faculty and student calendars, course Web sites, and electronic door keys to classrooms, and then generated e-mail notifications to everyone involved. While most systems are not yet able to achieve this level of integration, it is a goal that would solve persistent problems related to data accuracy.

The portfolio of software run by institutions often includes a variety of commercial, homegrown, and adopted software held together by various kinds of software "glue" that is developed locally or provided by vendors to help their products interoperate with other systems. Open source software such as Linux and Apache's Web server and suite of tools are commonly used, uPortal is widely adopted and incorporated into other software, and open source desktop applications such as OpenOffice are increasingly popular. These applications run side-by-side with commercially developed operating systems, databases, and a variety of homegrown applications and services developed to respond to local institutional needs. The risk of using these open source products is low because they are stable, there is a robust community of developers and support

providers, and it costs little to adopt them and then discard them when better tools come along.

As comfort with open source in IT portfolios grows through proven products and support structures, it's no surprise that community source enterprise applications such as uPortal, Quali, and Sakai are garnering the attention of university officials. While the licensing fees for commercial applications are often factored into decision making and the "free" nature of open source has appeal, chief information officers concur that the major costs of enterprise software are integration, support, and maintenance. These costs are common to all software, no matter the source.

## Evaluation Considerations

In some ways, there is little difference between the evaluation process for open source and commercial software. The software needs to meet the specified requirements, and if it does not, the difficulty and cost of bolting on the missing components must be assessed. The architecture of the software should fit the institution's technology and interoperability profile. Also, the experiences of others who have used the software, the available support options, and reputation of the company or community all factor into the decision-making process for both open and commercial products. Similarly, there is little difference in the postdeployment requirements between purchased, adopted, or locally developed technology. The institution must roll out the product, train the users, and support the environment.

There are, however, some differences between open and commercial software to consider. These are, in many ways, more illuminating than the similarities. Most obviously, costs are incurred differently between commercial and open source software. With commercial products like Banner, PeopleSoft, and Blackboard, costs are measured up front for licensing, then over time for maintenance and upgrade fees. Interoperability and integration between the product and other major systems is either imbedded into the licensing cost as inherent functionality or purchased separately through customization and localization contracts with the vendor. Annual license fees pay for software maintenance, and upgrades are typically purchased as they occur. If future upgrades require changes to the local customizations, there is an additional cost to recustomize the software.

With open source applications, the same issues of localization and customization must be considered. However, rather than purchasing these services through the vendor, an institution must either use staff time or purchase services from a third party. While no license and maintenance fees are involved, institutions adopting open source might invest more staff costs up front to accomplish the deployment. The upfront cost of open source includes time to become familiar with the code base as well as installing and integrating the software. Integration may be easier with open software because it can more easily be understood and modified, and because others who have accomplished the same integration may share their strategies and software as open source as well. For example, Indiana University has shared software to allow instructors to search their library holdings from within Sakai, automatically adding links in the Sakai sites so that

instructors do not need to manually copy and paste information as they build bibliographies. Other institutions are able to adopt this integration software.

Open source, unlike most commercial software, allows support to be decoupled from the software product, requiring a decision about whether to assume support in-house or contract with a third party. Rather than purchasing support directly from the software vendor, open source allows the selection of support from one of many third-party providers if the institution chooses to outsource application installation and/or maintenance. The models for support vary. For example, the Appalachian College Association has purchased hosting services for their Sakai-based course sites, while Foothill College contracts for professional services to support their production servers but manages their own software (Sakai, 2006). The benefit of third-party support is that the institution may select the support company that best meets its needs and may change support vendors if its needs are not met satisfactorily. The institution must maintain two relationships, though—one with open source community, and one with the vendor.

To make the right decision, administrators should evaluate the entire IT portfolio, seeking a synthesis that balances cost and product survival (Katz, 2006). While Katz refers to the context of relationships with commercial partners, it is equally relevant in the analysis of adding community source products to an institution's IT portfolio. Decision makers must balance the costs and support needs of the many products in their IT portfolios, seeking the right mix of functionality, interoperability, new products, and maintenance, while optimizing the available staff time and talent. Because each institution is unique, there is no single IT profile that is right for everyone. However, the decision factors of product evaluation, institutional values, and available staff skills are common to all.

## Product Evaluation

The three key considerations for selecting software are the features in the product itself, whether it will fit into the technical profile of the institution, and the costs to migrate or switch to the new product.

Functionality relates to the degree to which the product fulfills the business need. If no single product meets existing needs, an institution can customize a commercial or open source product, mix and match components from multiple products, or postpone the project until the marketplace catches up. Once a product replacement is under way, course corrections or a changed decision can be prohibitively expensive, which is why evaluation of product and architectural fit are critical.

The cost to maintain software over its life cycle is more than the cost of installation and deployment. Other factors to consider include:

- *Whether the products fit within the institution's long-range architectural plans and application profiles.* Variation in architecture can be costly. For example, if the institution's goals include implementing single sign-on for security, applications that do not provide intrinsic support for single sign-on must be

adapted with special interfaces. The cost of this work, as well as the cost of maintaining these customizations over the long term, must be considered.

- *Whether the products are modular and interoperable, use open standards, and allow interoperability for better management reporting and enhanced user experience.* The key question is, what will it take to make this system communicate and work effectively with the other campus information systems we deploy?
- *Whether each product is designed for higher education rather than being designed for commercial use and shoehorned into a higher education package.* The consequence of a poor product fit is extensive customization, leading to the worst of both worlds: expensive software to license and too much customization to upgrade. The institution will end up paying on both ends, with neither the fit and flexibility of custom-built software nor the leveraging of scale economies of commercial software.
- *Whether the product is so highly customized or uses unique data types as to create a barrier to exit at the end of its useful life.* When the next leap of innovation or capability is available in a few years, how costly will it be to upgrade or move to another product?
- *Whether the product runs on the institution's existing profile of servers, operating systems, and databases.* The cost of maintenance is an important consideration, and the ability to standardize across hardware and operating system software will help keep the total cost of ownership down. With rising energy costs, housing unneeded and redundant servers is an additional penalty.
- *Whether the product has matured.* If it is evolving, are the changes desirable?

Analysis of these criteria leads to an understanding of how much upfront and long-term investment will be needed to deploy and maintain the software.

## Core Values and Strengths

Assessing the institution's capacity and desire to undertake a project, its interest in cultivating expertise in technology and business process modeling, and its staffing infrastructure is as important as assessing the technical infrastructure. The values and core strengths of the institution, as they relate to IT, will vary with every project, particularly when viewed in the context of other projects. Schools with experience in deploying enterprise software can attest that the technology is not the only hurdle to overcome. Stanford's experience with Oracle Financials demonstrated that additional months of training and testing were required in order to make the project a success (Palmer, 2003). User practices, the maturity and adoption of business processes, campus readiness, and administrative commitment to the project are all key factors in product success or failure.

The institution's culture may be one of self-reliance and control in a particular business area, with a history of development and innovation, well-defined business processes,

and a core of professional staff with expertise. For example, one of the reasons cited by the University of California for joining the Kuali project was to leverage the expertise gained from developing a legacy system (Kuali Foundation, 2006). Or, the institution may not possess these attributes but may wish to build more self-reliance and expertise. In either case, undertaking adoption of open source software will fit the institution's culture. Alternatively, the institution may have a business need to meet but little interest in developing a core of internal expertise in this particular area. In this case a turnkey product or outsourced model will be a good fit for the enterprise application.

When deploying an enterprise application, whether commercial or open source, staff must have the expertise to define functional requirements, evaluate alternatives, and undertake the work. In addition to having sufficient resources, staff must be competent to evaluate and influence business processes and imbed the new application into the institution's workflow (R. Livingston, personal interview, April 7, 2006). Stories abound of institutions that outsourced business process and integration work to a vendor during a major deployment but failed to build internal capacity, leaving the institution without the skill base to maintain the application and the business processes over time. If the institution does not have core expertise, it does not matter whether the course management system is Blackboard or Sakai; in either case, deployment, support, and effective use of the product will be difficult. On a complex project such as a financial system deployment, limited core capacity for this work can doom a project to delays, poor decisions, and budget overruns. While core capacity in business functions is necessary to deploy any enterprise system, adopting open source adds one additional consideration. Someone needs to have sufficient technical skills to work with the source code. Of course, this work can be outsourced to a third-party integrator, and many institutions have successfully adopted open source by working with commercial partners. Whether outsourced or undertaken internally, the same question applies—is it important to the institution to have these skills, and if so, how will they be assured?

Finally, the institution should consider its interest in influencing the direction of the product. If internal development is under consideration to create an application that meets the institution's needs, then joining an open source project is a cost-effective alternative for ensuring local needs are considered. If the institution's requirements are a part of the product, then new versions will not need to be recustomized, saving money in the long run. By working in the community, the institution can influence product direction while leveraging the scale economies gained by working with others.

### Company or Community?

When making a buy-or-adopt decision, an institution is entering into a relationship with a company or a community. The relationship will likely last for years, certainly through the first product life cycle and potentially through many versions and upgrades. Selecting a product is not only a matter of technical evaluation but also of due diligence on evaluating the company or the community. The size and complexity of the project will dictate whether it is feasible to consider shifting the technology in a few years or if the nature of the investment necessitates a longer investment. In either case, it is important

not to align with a vendor or community too early in the decision-making process but to take time to evaluate the product and motivation of the person pushing the product.

Carl Jacobson, Director of Management Information Systems at the University of Delaware, commented, “Nothing about the fact that a product is commercial makes it a panacea of good” (C. Jacobson, personal interview, April 21, 2006). Technology is a dynamic field with regular entrances and exits, mergers and acquisitions, and evolving alliances. Higher education is a small segment of the overall technology market, and the needs of the academy may not be primary to a company. To be sure, many companies focus on higher education, giving care to the needs of colleges and universities and aligning their products with the academy’s business needs. In other cases, the concerns of higher education are a low priority.

Vendors have long said, “Give me your money and I’ll make your problems go away.” We know this hasn’t worked, and after a decade or two of enterprise deployments, university administrators are skeptical (R. O’Leary, personal communication, May 10, 2006). Open source projects now offer a similar promise, asking for resources and commitment in exchange for a product and a community of peers, sometimes in the early days of a project while the first generation of software is still under development. Given past experiences with expensive ERP deployments, it is not surprising that administrators are exercising caution in developing new relationships with companies or communities.

An institution that is evaluating an open source strategy should also consider the roles it wishes to play. Does it want to be an adopter only, or should it participate in the project? Either is an acceptable strategy. Adopters will work with the code base, performing local integration and perhaps collaborating with other members of the community on best practices. Contribution is similar to purchasing a product in that resources are devoted, but rather than licensing and maintenance fees, staff time is exchanged. Software contributions are common but are not the only form of participation; providing documentation, user support expertise, product testing, and collaborative infrastructure and contributing local adaptations are equally important to a community-based effort. For example, Texas State University–San Marcos has contributed its technology, documentation, and best practices for migrating from its legacy course management system to Sakai.

As a contributor, an institution must have business knowledge and technical skills to make a product better as well as a commitment to participating in the project. Stanford’s experience with the Sakai Project demonstrates this situation. Faced with a legacy course management system that needed to be replaced, Stanford opted to join Sakai rather than undertake another iteration of locally built software. Sakai offered a way to preserve Stanford’s expertise gained through years of research and development while breaking away from homegrown software. Stanford has contributed design, best practices, and approximately one-fourth of the software in Sakai, representing significant amounts of staff time and expertise. In turn, Stanford has received three times the amount of software it contributed, as well as others’ expertise in software design and user support. It “costs” more to contribute to than to simply adopt open source software,

but the rewards include influencing the product direction, increasing staff skills through work in the consortium, and helping ensure the success of the endeavor, thereby protecting local investment.

## What It Means to Higher Education

An institution considering working with open source software has several entry strategies that offer nontraditional approaches to deployment:

- Applications can be adopted and deployed for a low-risk pilot in an area not currently supported. For example, many institutions are working with the Sakai software in pilots to support research collaboration, giving them a chance to gain familiarity with the software and the community before considering replacing their course management systems. Similarly libraries are testing DSpace and Fedora for repositories alongside more traditional library systems rather than replacing existing functionality.
- Pieces of open source software can be mixed and matched with other applications. For example, uPortal is being used to offer an entry point for administrative applications, while Quali Financial System and Quali Research Administration have modules that can be mixed with other systems. For example, Quali Enterprise Workflow manages most data changes for PeopleSoft HR at Indiana University.
- Small schools are working in consortia to share instances of software, spreading the cost of ownership over several institutions. Similarly, several vendors offer ASP and test-drive services so that institutions can gain familiarity with feature sets without having to invest in installation and local knowledge.

In a dynamic marketplace, the choices for enterprise and infrastructure software are complex and constantly changing. While these applications may be a requirement for business processes at an institution, they often do not directly support the core missions of learning, teaching, and research. Further, they are expensive to deploy and maintain. While enterprise applications are a major investment, the level of turmoil associated with their deployment needs to be minimized. An institution needs to install the application, have it work well, and realize adequate return for the investment.

## Key Questions to Ask

- To what degree does our institution have the technical resources to devote to a community-developed software application? For the product we are considering, what are the options for participation?
- Does our institution have a pressing business need that must be met quickly, are we extending services, or are we looking for a low-risk pilot to learn more about a product?

- What are the entry and exit costs for extending or adopting our preferred commercial product? What are the equivalent costs for community source software?
- What benefits will accrue to the institution from purchasing a commercial software product? What are the benefits of participating in a collaborative development project?
- Does the company or community have a core competency in developing good quality software, using open standards to allow interoperability?

## Where to Learn More

- Dodds, T. (2006, July/August). Changing assumptions in best practice. *EDUCAUSE Review*, 41(4), 76–77. Available from <http://www.educause.edu/LibraryDetailPage/666?ID=ERM06410>
- University of Texas at Austin Information Technology Services. (n.d.). Thinking about buy vs. build. Available from <http://www.utexas.edu/its/eis/buybuild/case/index.html>
- Wheeler, B. C. (2003). *Aligning IT strategy to open source, partnering, and Web services*. Boulder, CO: EDUCAUSE Center for Applied Research. (Research Bulletin, Issue 24). Available from <http://www.educause.edu/ecar/>

## Acknowledgments

The author thanks the following individuals for sharing their insights about open source and higher education: Richard Holeyton, Senior Strategist for Student Computing and Associate Director, Academic Computing, Stanford University; Carl Jacobson, Director of Management Information Systems, University of Delaware; Randy Livingston, Vice President for Business Affairs and Chief Financial Officer, Stanford University; Christopher J. Mackie, Associate Program Officer, The Andrew W. Mellon Foundation; and Robert O’Leary, Executive Director of Administrative Systems, Stanford University.

## References

- Abel, R. J. (2006, March 1). *Best practices in open source in higher education study: The state of open source software*. Lake Mary, FL: Alliance for Higher Education Competitiveness, Inc. Available from [http://www.a-hec.org/open\\_source\\_state.html](http://www.a-hec.org/open_source_state.html)
- Katz, R. N. (2006, January/February). Honey, have you seen my market? *EDUCAUSE Review*, 41(1), 76. Available from <http://www.educause.edu/LibraryDetailPage/666?ID=ERM06111>

- Kualifoundation. (2006, July). University of California invests \$1M in KFS and joins as Kualifoundation Partner. *Kuali Progress*. Available from <http://kuali.org/newsletter2006-2>
- Palmer, B. (2003, October 4). Delphi rollout set for September after postponement allows for testing, training. *Stanford Report*. Retrieved August 5, 2006, from <http://news-service.stanford.edu/news/2003/january29/delphi-129.html>
- Sakai. (2006, May 27). Appalachian College Association selects Longsight for Sakai services. Retrieved August 5, 2006, from [http://www.sakaiproject.org/index.php?option=com\\_content&task=view&id=397&Itemid=312](http://www.sakaiproject.org/index.php?option=com_content&task=view&id=397&Itemid=312)

## About the Author

Lois Brooks ([lbrooks@stanford.edu](mailto:lbrooks@stanford.edu)) is the Director of Academic Computing at Stanford University.

Copyright 2007 EDUCAUSE and Lois Brooks. All rights reserved. This ECAR research bulletin is proprietary and intended for use only by subscribers. Reproduction, or distribution of ECAR research bulletins to those not formally affiliated with the subscribing organization, is strictly prohibited unless prior permission is granted by EDUCAUSE and the author.