EDUCAUSE Center for Applied Research

Research Bulletin

Volume 2003, Issue 24

November 25, 2003

Aligning IT Strategy to Open Source, Partnering, and Web Services

Bradley C. Wheeler, Indiana University

4772 Walnut Street, Suite 206 ◆ Boulder, Colorado 80301-2538 ◆ www.educause.edu/ecar/

Overview

Evolutionary shifts in technology, user requirements, and the economics of information technology periodically create inflection points for reassessing IT strategy. The current inflection point recognizes that widely available IT technical standards (e.g. those from the Open Knowledge Initiative (OKI)[1]) and data standards (e.g. from the IMS Global Learning Consortium[2] are making open source's "borrow" an economically-necessary addition to the classic build versus buy decision for acquiring application software.

In general, open source means that the source programming code of the software is available to the public, with certain rights and limitations regarding its use. Access to the source code allows the adopter to deeply understand or alter its functionality. There are many questions regarding what value, if any, open source brings to higher education. Some view open source as the rallying point for an anti-vendor philosophy, while others may see it as a highly risky venture in terms of quality and support. Some see it as aligning with the core university value of openness and knowledge sharing, yet others fear that it robs the university treasury of potential licensing revenue. While the practice of open source software is decades old, it merits renewed scrutiny as a viable means to achieve much needed economies in software development.

The term "borrow" is used to convey the general spirit of the open source philosophy as adopters of the software may contribute to furthering its features or debugging it. Unlike borrowing from a bank, however, there is no formal responsibility to give back because broad adoption of the software by many is viewed as a signal of success in the open source community.

The real value of open source application software for higher education, however, is unlikely to be captured outside of a disciplined IT strategy that recognizes "code mobility" as a viable new path to shared economies. This research bulletin examines the current state of open source initiatives in higher education, and it provides a roadmap for incorporating open source application development into an IT strategy. The bulletin addresses internal strategy alignment, inter-institutional partnering arrangements, intellectual property licensing, and the evolution of application software as web services.

Highlights of IT Strategy Alignment

The essence of IT strategy is making choices, and the constantly co-evolving worlds of IT capabilities and institutional needs impose many choices on IT leaders. These choices are often situated in the path dependencies of prior choices (e.g. existing systems), and, more importantly, they will project path dependencies onto future choices. The overall coherence of many choices over the years is well served by a disciplined and purposeful IT strategy.

Early thinking on IT alignment was framed as simply aligning IT structure and investments to the institution. For example, highly decentralized institutions in terms of finance and authority would find constant misalignments with highly centralized IT

control. Alignment would favor decentralizing the IT organization. Institutions that demonstrate strong values for cost containment would view highly efficient IT with little investment in pioneering innovation as more aligned than investments in a series of leading edge, speculative IT pilot projects. Such a view of alignment, however, would fail to grasp the potentially transforming opportunities in IT-enabled business and educational processes.

Current views of IT alignment recognize the need for co-evolution and mutual adjustment of IT and institutional strategy in the context of changes in the external environment. The open source software option has matured as an important element in an IT strategy that is co-evolving with the dual institutional demands of delivering innovative IT services while managing IT for value.

Application Software: Build, Buy, or Borrow?

One element of IT strategy is sourcing software—particularly for large-scale application software such as course management systems, financial systems, or library systems. The complexity of these systems makes their creation a large fixed cost with a potential for significant on-going maintenance costs. The traditional choice has been between building (developing in-house) or buying (licensing from a vendor) these systems with each choice imposing a distinct set of tradeoffs affecting IT alignment (see Table 1). Because all systems have hardware and often substantial implementation/integration costs—irrespective of the sourcing option for the software—those costs are not discriminators in a sourcing strategy.

Table 1. Build, Buy, or Borrow Matrix

	Fit with Requirements	Acquisition Cost	Maintenance Cost	Support Options	Control of Destiny
Build	▪Tailored to requirements	▪Full cost ▪Expensive permanent staff or contract	▪Discretionary ▪Full costs for changes ▪No ongoing fees	▪Institution	▪Very high ▪Own the code
Buy	▪Standardized ▪Tailored via add-ons	▪Shared cost, plus vendor profit as license fee	▪Mandatory ▪Shared cost, plus vendor profit as license fees	▪Vendor(s) ▪Warranties and service-level agreements	▪Very low ▪Limited/no access to modify code ▪Extensive add-ons may complicate upgrades
Borrow	▪Assembled from standardized components and tailored	▪Nil, minimal, or shared cost	▪Discretionary ▪Nil, minimal, shared, or full cost	▪Institution ▪For-fee vendors ▪Partners ▪Community	▪Very high ▪Full access to the source code

Over the multi-year life of software investments, a sourcing strategy must balance attributes of the fit with institutional and user requirements, acquisition costs, maintenance costs, support options, and control of destiny over time. Answers to these questions for a single application, such as a library system, could look different when the system is considered within a portfolio of related university systems over multiple years.

The build or buy approaches are well understood. An IT strategy that favors building software may be well suited to a large university that has economies of scale in leveraging developers, maintenance, and project management across many systems. Likewise, building may be economically infeasible when economies of scale in expensive staff are not sustainable. Building gives great control over software functionality, minimizes ongoing maintenance and license fee obligations to vendors, and provides full control of options and timing for a system's destiny over the years.

An IT strategy that favors buying vended software benefits from the shared economies of scale created by a vendor. Such a strategy can also simplify the administrative agenda by not managing a large software development shop over the years. The vendor aggregates demand for a system, manages the development and maintenance, and distributes the cost plus a profit margin in license fees across a number of customers. This often provides quick access to very high quality software at (arguably) less than the initial cost of building it. The destiny of the system, however, depends on the path and timing chosen by the vendor, and this path becomes more complicated as the number and diversity of customers increase.

Open source software provides a blended model with attributes from both the build and buy approaches. First, open source can provide immediate, low- or no-cost access to software code, though implementation and other costs make this far from a free approach to systems. Second, the open source code may suffice as a packaged solution like the buy approach, or it may provide a jump start for further local coding to produce a tailored system. Maintenance can be handled internally, contracted to a for-fee vendor in some cases, or can rely on a broader open source community that addresses maintenance issues. Third, there are no ongoing licensing fees, leaving the destiny of the system—for better or worse—in the hands of the institution or the community.

Historical Barriers to Code Sharing

The concepts of open source and application software code sharing among higher education institutions are not new. Despite much hope, however, a number of real and perceived barriers have impeded widespread code sharing within higher education. These obstacles include

- "But my campus is different"
- Timing differences for software investments among institutions
- Differing technology architectures
- Staff attitudes regarding open source quality—"not invented here"

- Licensing and intellectual property
- Support distraction fears at software-developing institutions
- Support availability fears at software-adopting institutions
- Sourcing decisions absent a coherent IT strategy

If higher education is viewed as an industry, these barriers have been an extraordinary drag on its collective productivity. It is difficult to argue the value of investing scarce institutional funding to "reinvent the wheel" by building a new software application when it has already been largely done elsewhere. Recent developments argue that now is the time to reexamine the veracity of these barriers.

What Has Changed

Four trends provide new reasons to consider code sharing via open source in higher education. First, the open source approach has demonstrated its viability in both the infrastructure and application software domains. The open source Apache Web server and its related services deliver a robustness and economic advantage against many commercial rivals for some Web serving requirements. Open source applications such as uPortal have demonstrated code mobility and viable community/vendor support in higher education.

Second, software applications are becoming more modular as web services and less monolithic. Services-based portals that deliver personalized access to university services and data (e.g., pay my bill, find a book, retrieve my assignment grade, etc.) will continue to pressure monolithic software applications to evolve as an unbundled collection of web services. For example, a course management system designed as a collection of web services can continue to be accessed as a cohesive system, or its functions (e.g., course calendar, discussion tool, etc.) can be rendered as services in a personalized portal. Portals can provide much of the user interface and navigation to further reduce the size of software components. Code mobility in higher education is accelerated because components (e.g., a grade book application or library search tool) can be adopted and integrated more easily than an entire system from another institution.

Third, the very real technical barriers of heterogeneous local hardware and architecture choices are being overcome by separating application software from these local choices. The OKI's Open Service Interface Definitions (OSIDs) provide a layer of technology insulation between the unique choices of a local institution and the common connections to an application.[3] As such, OKI-based applications can move quite easily among institutions that have implemented OKI services. An e-portfolio application that needs to authenticate a user does need not care that one institution uses CAS and another uses Shibboleth. Similarly, common data specifications based on eXtensible Markup Language (XML) from standards bodies such as IMS are mitigating the proprietary data formats of application systems. These specifications and emerging standards are accelerating application software code mobility among institutions.

Finally, leadership and staff attitudes toward open source software are becoming more favorable. Perceptions of shoddy quality or no support are giving way to the realities of viewing open source as a viable set of tradeoffs alongside building or buying. Economic realities, however, may be the real factor that furthers open source development where there is a large, shared need among many institutions and the political will to partner.

What Open Source Means to Higher Education

The four trends described above mean that new economics are accessible to higher education—perhaps for the first time with real credibility. The timing couldn't be better, given the recent cost scrutiny of the industry and pressures for IT innovation.

Code Mobility

Code mobility is the key economic bet for higher education application software. Higher education has neither the time nor resources to constantly reinvent highly similar applications at various institutions, and vended software should not be the only alternative because it also carries risks. For example, a series of reports in the *Chronicle of Higher Education* documented how low, up-front prices helped build market share, followed by a predictable software industry consolidation over the years. Institutions then faced high switching costs once software was implemented, and oligopolistic pricing naturally followed as the remaining vendors worked toward sustainable profitability.[4]

Similarly, many institutions custom built their own portals, course management systems, library systems, and so on, and then individually (and redundantly) bore the costs of maintaining and enhancing those applications without access to shared synergies among institutions. Tight integration in local architecture and other technical challenges often made sharing these local applications infeasible.

The payoff of managing a disciplined IT strategy to leverage code mobility will become greater as web services and interoperability frameworks become more widely adopted in higher education. Frankly, code mobility is higher education's best bet to deliver on IT alignment's dual challenges of delivering innovative IT services while managing IT for value. A disciplined IT strategy that leverages the open source option, however, will not emerge on its own. Engagement with open source requires purposeful leadership actions.

Leadership Actions

The first role of any IT leader is to deliver required functionality from IT systems, but this must be balanced and executed within an economically sustainable approach. Thus, IT leaders who believe that open source is an important economic part of their future can begin to steer internal discussions through establishing application sourcing principles, understanding models of participation, considering the role of institutional timing, and steering institutional policies for open source licensing.

Application Development/Sourcing Principles

One example of an IT strategy that provides guidance for application development is presented in Table 2. These principles are not used as a straightjacket for sourcing decisions, but they do illuminate the default path as many hundreds of small and large sourcing decisions are made across an institution.

Table 2. Example Sourcing Principles

Principle	Example
Standards	We will enhance our opportunities for code mobility among universities by architecting on a common layer of OKI services as our baseline infrastructure for new our applications. The complementary data standards will be based on data specifications from widely accepted higher education groups (such as IMS, etc.) whenever applicable. J2EE, AIX/Linux, and Oracle are the standards for enterprise-scale application development.
Sourcing	For systems developed in-house, we will participate, whenever possible, in open source approaches—both in importing existing solutions and in exporting our solutions. We will partner with like-minded institutions whenever goals and resources align to share costs.
Delivery	We will focus on personalized delivery of information services and activities via the personalized portal through an unbundled, web services approach to application development.
Leverage	We will aggressively seek efficiencies in consolidation of redundant application services via leverage of web services whenever feasible.

Participating in Open Source

There are many paths to open source application software, and some areas will evolve a vibrant community similar to the Apache or uPortal successes. Four models stand out today (Table 3), and features of each can be matched to a particular opportunity. Participants in each model hope their projects will evolve into sustainable communities that maintain the application.

Table 3. Open Source Development Models

Model	Features	Examples
Lead Institution	▪Institution takes lead in writing an application for its own needs ▪Develops for code mobility using a framework/standards ▪May lead a community that becomes more of a consortium model over time	▪CHEF Project—University of Michigan
Partnering	▪Formal or informal agreements among a small group of institutions to write tools ▪Tools integrate as part of a planned application framework	▪Navigo Assessment Project—Indiana University, University of Michigan, Stanford University ▪Fedora—University of Virginia, Cornell University
Consortium	▪Extra-university entity that coordinates application requirements, standards, and releases ▪Coordinates a community	▪uPortal—JA-SIG ▪ePortfolio Project—Open Source Portfolio Initiative ▪Chandler Project—Open Source Application Foundation ▪SAKAI Project
Consumer	▪Institutions or vendors that implement open source systems with minimal/no participation in its development ▪Waiting to adopt code from others	▪Any institution that downloads and implements open source application software ▪Most institutions will consume open source code for some needs because that is part of their sourcing strategy

The Lead Institution model has the largest up-front cost to an institution, but it also affords the greatest potential payoff in influence of an application, its standards, and pioneering recognition. Partnering is best suited to managing risk for a specific project of limited scope and duration. Partners can be selected for similarities that provide a common basis for perspective, timing, and commitment to success. Consortia have proven a useful means of aggregating demand, garnering external support from foundations, and delivering a specific project for broad adoption. All provide some mitigation of risk over a purely vended option because the source code is available to all should devastating circumstances intervene. Almost all institutions—both those that contribute open source code and those that do not maintain a development staff—will consume some open source as part of their IT strategy.

Synchronizing Institutional Clocks for Leverage

A third leadership action involves finding opportunities that provide natural inputs for shared open source development. Various institutions often mobilize their institutional energy to develop applications at different times. For example, a need for e-portfolio software may be a pressing priority at the University of Minnesota at a time when it is only beginning to gain attention at Indiana University. Institutions that accelerate or slow

their software development or implementation clock to match other like-minded institutions can grab the economies afforded by partnering or by one of the other models described above. Synchronizing development and implementation around a project can pay continuing benefits in later-year shared maintenance and shared software enhancements as innovation.

Resolving Licensing Matters

Finally, IT leaders must understand and influence institutional policies regarding contributing internally developed software into the open source community. This is a non-trivial matter for many institutions' technology transfer offices—especially if they view all university intellectual property as having potential commercial value. A quick review of <http://www.opensource.org/> will show great variance in open source licenses.

At the heart of this is "open-open" licensing that allows anyone to use or to commercialize without fee or restriction. An open-open license, such as the Navigo Project, OSPI's ePortfolio, or uPortal, allows for-profit vendors to package, modify, or sell the software without any royalty fee obligation to the original copyright holder.[5] The role of these commercial partners—providing code enhancements, bundled products, fee-based consulting, hosting services, or integration for institutions that need these services—has proven an essential element in establishing a sustainable open source community for a particular application.

In contrast, some institutions prefer an "open-closed" license for their intellectual property (IP) that allows free use of IP for educational purposes, but restricts commercial modification and sale of the software to a negotiated licensing agreement with the copyright holder. This non-commercialization restriction virally infects any derivative software that is built upon or extends the original work because the derivative work would also be subject to the licensing negotiation with the original copyright holder. The restriction effectively drives away vendor investment in sustaining a product, service, or its open source community. An open-closed licensing policy is a bet that an institution can freely share the software among educational institutions while impeding vendors' incentives to become part of the application's open source ecosystem. Of greater concern, however, is that an open-closed licensing policy limits an institution's ability to participate in open source partnering because many other institutions shun the viral licensing practice.

Conclusion

Experienced observers of higher education collaboration initiatives could reasonably expect the code mobility bet to fail. The reasons to be optimistic, however, are that open source can be an effective tool in addressing the dual challenges of managing IT for value while delivering innovative software. Institutions' views on open source and partnering as part of a software sourcing strategy are collectively self-fulfilling in higher education. If higher education chooses to invest its significant development and administrative resources in code mobility, then the bet is it will succeed. If institutions

view code mobility as only a hedge bet alongside a core buy or build IT strategy, then open source's compelling economics for higher education will remain unrealized.

Key Questions to Ask

- Does my institution have an IT strategy that provides guidance for software sourcing tradeoff decisions?
- Why would my institution want to engage open source?
- Which model of open source participation is best for a particular need?
- What institutional changes are required to engage in open source?
- Will we contribute, consume, or both?
- What would define successful engagement with others (partnerships, etc.)?
- Can we contribute our work and intellectual property under open-open licensing?

Where to Learn More

- Open Source, <http://www.opensource.org/>.
- Buy Versus Build, <http://www.utexas.edu/its/eis/buybuild/>, and the Common Solutions Group September 2003 meeting sourcing workshop, <http://www.stonesoup.org/>.
- The SAKAI Project, <http://www.sakaiproject.org/sakaiproject/>.

Endnotes

1. The Open Knowledge Initiative, <http://mit.edu/oki/>.
2. IMS Global, <http://imsglobal.org/>.
3. Open Service Interface Definitions, <http://mit.edu/oki/>.
4. F. Olsen, "Getting Ready for a New Generation of Course-Management Systems," *Chronicle of Higher Education*, December 21, 2001, A25; "Pricing Changes by Blackboard and WebCT Cost Some Colleges More—Much More," *Chronicle of Higher Education*, March 19, 2002; "Course-Management Companies Are Still Seeking Elusive Profits," *Chronicle of Higher Education*, May 23, 2002.
5. See <http://www.navigoproject.org/> for an example of an open-open license. See also <http://www.theospi.org/> for ePortfolio license and commercial support (http://www.rsmart.com/), or <http://www.uportal.org/> for uPortal license and commercial support by various companies including Unicon/IBS, <http://www.interactivebusiness.com/ >.

About the Author

Bradley C. Wheeler (bwheeler@indiana.edu) is the Associate Vice President for Research & Academic Computing and Dean of IT at Indiana University. He is also an Associate Professor of Information Systems in the Kelley School of Business.