The Open-Source Movement

Despite many obstacles, open-source has the potential to strongly influence the future of software development and support in the academic world

By Thomas Warger

The open-source movement is sur-

facing more and more often as an un-
dercurrent in the busy flow of discus-
sion swirling around software de-

velopment in higher education. Most of-
	en often it comes up for mention as a re-

sponse to the increasing predomin-

ation of commercial, proprietary soft-

ware in use on campuses. As operating

systems, development tools, desktop

applications, and enterprise software

all have become large, complicated, and

expensive, an increasing number of IT

professionals are looking for not just

alternative products and sources, but

at a different way to develop and sup-

tort software. If open-source fulfills its

proponents’ hopes to even a modest
degree, the effect on IT practices in

higher education will be substantial.

Open-source can be defined as an ap-

proach to software development and in-
tellectual property in which program
code is available to all participants and
can be modified by any of them. Those
modifications are then distributed back
to the community of developers work-
ing with the software. In this method-
ology, licensing serves primarily to dis-

close the identities of all the partic-

pants, documenting the development

of the code and the originators of

changes, enhancements, and derivative
derivative off-shoots.

The most widespread and vocal adher-

ents of open-source are the members

of the Linux-using community. But pro-

jects sponsored by major universities
to develop new “open” software are

also underway. The most visible of these
is the Open Knowledge Initiative, a con-

sortium of American universities led by

MIT and Stanford. Their aim is to pro-
duce an “architectural specification”
for the development of educational soft-

ware. The Java in Administration Special
Interest Group is a large association of
academic and commercial organi-

zations sharing Java code and collabor-

ating in the develop-

ment of uPortal, an open-source
campus portal product.

For the most part, the open-

source technologies and

products existing or

under development
today are not primar-
ily unique or ground-

breaking in func-
tionality. Instead, they are alternatives
to commercially

well-established

software, distin-

guished more by the way they are

owned, operated, and further de-

veloped. A college or university buying

a commercial portal or operating system
agrees to license terms and conditions
that almost always prohibit any modi-
fying of the software. The software itself
comes only in compiled form and so is
not amenable to being changed in any
event. Frustrations with those con-

straints are the basis for interest in open-

source.

The Proprietary Grip

Information technology on campus
has settled into a pattern of relying on
commercial, proprietary soft-

The Edutech Report is a monthly, subscription-based publication of Edutech International (http://www.edutech-int.com), covering the wide range of issues in higher education information technology. Copyright 2002 Edutech International. All rights reserved.
As long as commercial software appears more affordable than programming staff, it will remain dominant.

Cost and License

Another consequence of the shakeouts that follow the rise of several competing, differentiated products and companies is the increase in fees and tightening of license terms and conditions. This fact of the marketplace has held true for all kinds of software: office suites, library packages, courseware management systems, and ERPs [enterprise resource planning systems] among them. The trend for total costs of ownership for software [is] upward, and more sharply where competitive pressure among vendors declines.

Skills You Need

Control over software, whether at the level of operating system, development tools, or application, has a strong psychological appeal to IT professionals. There are, however, strong practical barriers to stepping up to that control. One of the reasons that commercial, standard software replaced locally, purpose-written products was that the sheer volume and complexity of program needs overwhelmed the skills of IT staff at many institutions. The reality of the academic IT scene is that relatively few institutions have the on-staff skills to develop software. And even where that capability exists, it is reserved for a few strategic projects. Almost nobody believes today
that custom writing is the best approach for the general run of software needs. Consequently, the realistic prospect of substantial software development activity is limited to projects with special resources or an unusual willingness to take risks.

**Open Is Not Free**

The biggest cost in IT is personnel — the time and talent needed to make technology work. Colleges and universities have adjusted their budgeting practices to accommodate the need to sustain the hardware base, which at one time appeared an insurmountable cost. More recently, the cost of AIS software became the new focus of concern. But in the background, the size and skill sets of IT staff under the funding limitations common in the academic world have proven a more fundamental shaper of IT working methods.

The acquisition cost of software is only just the beginning. Because open-source code is available free or at very low cost, there is a temptation to think that cost savings will fund a lot of development work before the balance begins to tip against open-source. The problem in this thinking is that while fast progress can be made in prototyping and initial development, the longer — and therefore more expensive — phase of work still lies ahead in the detailed programming and hardening of the software. As a result, the cost advantage of an open-source solution depends heavily on avoiding the crushing burden of systematic programming.

**To Build Systems**

The history of software writing has been the transition in outlook from writing code units to engineering complex systems. Organizations specialized to work in this manner have all but taken over the development of software. The question prompted by this trend is whether a campus IT organization, even if supplemented in its efforts by consortium or commercial partners, can be effective in this role.

The likelihood that institutions will reverse the exit that most of them made from primary code writing appears very small. Few senior administrators will be convinced to return to in-house development of software, especially given the on-going difficulty of supporting campus IT needs and ambitions as they currently stand.

**Focus on Tools**

Inside the IT organization, one of the big potential benefits of open-source is a new focus on software tools. Linux/Unix, Java, PERL, and SQL — which are not yet in the skill sets of many IT staff — have two types of value that have been only sporadically exploited. They can be used to create valuable new capabilities in the Web environment that is the focus of so much backlogged demand. They also provide the foundation for a new level of self-confidence to consider local and immediate responses to a wider range of applications needs.

Open-source utility software is already the basis for sharing across a wide range of IT organizations. By training staff to use these tools, campus IT groups would also be promoting connections with the wider IT community, where open-source solutions are more common than among staff trained on counterpart, proprietary tools.

**Using the Web**

The greatest benefit in open-source could be the opportunity to realize the best promise (and original purpose) of the Web: to make an extended working environment where information is accessible to all those involved in collaborations. The prospect of very large communities focused on shared projects offers an intriguing alternative to the prevailing “industrial” model of software development, where a single, formal organization specialized for production is currently the rule.

Very little is known about how this 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.

Contributed code libraries are one of the oldest features of the computer era. The big question in open-source is whether it can lead to a new way to organize work on software.

---

Thomas Warger (twarger@edutech-int.com) is Managing Editor of The Edutech Report.