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 surfacing more and more often as an undercurrent in the busy flow of discussion swirling around software development in higher education. Most often it comes up for mention as a response to the increasing predomination of commercial, proprietary software 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 support software. If open-source fulfills its proponents' hopes to even a modest degree, the effect on IT practices in higher education will be substantial.

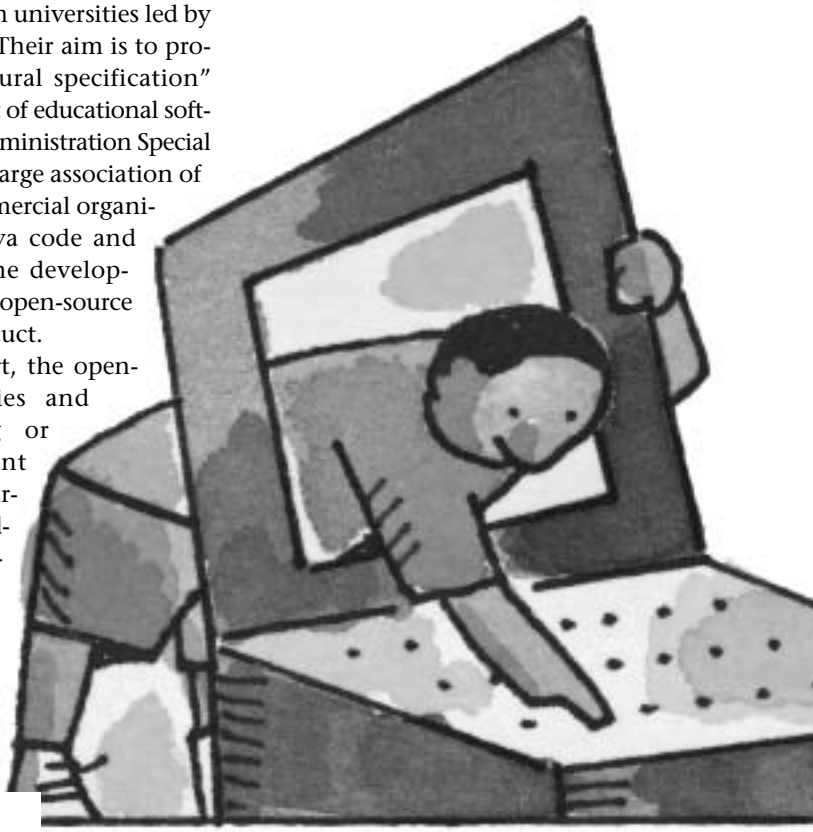
Open-source can be defined as an approach to software development and intellectual property in which program code is available to all participants and can be modified by any of them. Those modifications are then distributed back to the community of developers working with the software. In this methodology, licensing serves primarily to disclose the identities of all the participants, documenting the development of the code and the originators of changes, enhancements, and derivative off-shoots.

The most widespread and vocal adherents of open-source are the members of the Linux-using community. But projects sponsored by major universities to develop new "open" software are also underway. The most visible of these is the Open Knowledge Initiative, a consortium of American universities led by MIT and Stanford. Their aim is to produce an "architectural specification" for the development of educational software. The Java in Administration Special Interest Group is a large association of academic and commercial organizations sharing Java code and collaborating in the development 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 primarily unique or groundbreaking in functionality. Instead, they are alternatives to commercially well-established software, distinguished more by the way they are owned, operated, and further developed. A college or university buying a commercial portal or operating system agrees to license terms and conditions that almost always prohibit any modifying 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 constraints 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-



ware. Computer, server, and network operating systems were the first to follow this trend, although since 1969, Unix — and more recently Linux — have remained significant exceptions to the rule. Commonly used desktop applications followed suit.

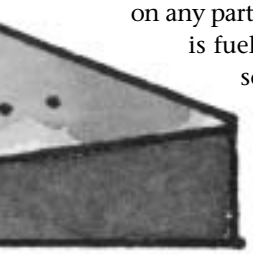
Later to follow were the administrative applications systems — what we currently call administrative information systems (AIS) software. Many utilities, including those for page definition, data transfer, and media streaming sprang into the world as virtual black boxes, their inner workings hidden from those who use them. A kind of backlash has set in and finds sympathy if not universally strong support from many in the IT community.

Technical objections to the essentially closed nature of most of the software now in use seem to grow stronger with each major new release of the major proprietary operating systems. The voices of technical support staff are, however, all but lost between the raucous promotion from the vendors and the opinions of end users and the popular computer press. With each new release of operating systems software, the IT world waits passively to be reshaped by the vendors' decisions about what the technology will be.

Each new generation of AIS proves extraordinarily more difficult to install, largely because of complexity of two kinds: the accumulation of local processes, exceptions, and customizations on the one hand and the rapidly growing set of options carried in the basic AIS packages on the other — and it is the second of these that seems easier to blame. These systems were once welcomed as a great advance over home-written applications, but there is now a growing perception that they are getting bloated and out-sized.

Fitness

Concern over the fit of commercial software with needs as they are viewed on any particular campus is part of what is fueling interest in open-source solutions. A crisis of confidence is building because it is never clear who



As long as commercial software appears more affordable than programming staff, it will remain dominant.

demanded the rash of new features that bulk up each new release of commercial software, a pattern just as true for desktop productivity suites as for enterprise software. Almost inevitably, the conclusion tends to be that those making the decisions in development of these products are out of touch with real needs or too busily trying to meet a range of needs.

As software companies consolidate, with a few dominating where there was once more competition, they tend to expand the scope of their products, feeling the need to provide solutions for all segments of their market. Selective focus and innovation are strategies for emerging companies. Mature, successful companies wage an all-fronts defensive battle against upstart innovators by adding features at the same time to many aspects of their product, but leave their users less satisfied with the fitness of those products for their needs. For these users, open-source looks like a return to basics, or at least an approach driven by expansion for its own sake.

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.

Software costs have taken over from hardware the dubious distinction of being one of the hard-to-control aspects of IT. So far, most institutions still find staff costs (and the difficulty of hiring adequate technical talent) greater challenges than the rising cost of their standard and familiar software. As long as commercial software appears more affordable than programming staff, it will remain dominant. But if the balance shifts — through increasing license costs and constraints, for example — open-source is likely to gain.

Control

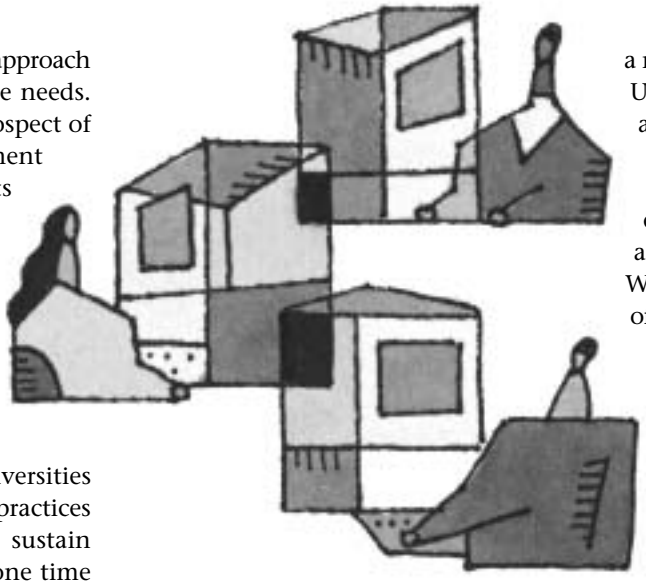
Technical staff increasingly serve as the maintainers of commercial software packages. They had no role in developing most of the applications they support and have little ability to change them substantially. As a result, those staff are perpetually caught between demands of the users they support and the vendors who supply the software and retain control over its functionality, shortcomings, and future evolution.

One of the consistently alluring promises of open-source is return of control. As frustrations with the locked-down character of commercial software grows stronger, the open-source idea seems to provide a way of regaining the lost control. Recognizing this line of reasoning, some software vendors have offered to make their products easier to extend and enhance. Blackboard and WebCT, for example, made statements to that effect when the Open Knowledge Initiative project was announced.

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.

Influence

Still, despite all the obstacles, open-source has the potential to strongly influence the future of software development and support in the academic world. There are already a few signs that the combination of IT professionals' frustrations and open-source alternatives are making major software makers think about opening their code to the development community.

Nothing in principle prevents software companies from transferring code into the open-source realm. The key question is whether their competitive advantage will be better served by a model of co-development with their client community or by trying to meet all needs by continuing the closed model. To the extent that commercial software companies believe that open-source is a viable option, they will be influenced to allow users more control over the software, at least in the ease of extending and customizing applications.

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. *e*

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