Learning Management Systems and the Realities of Using Open Source Software

PHILIP ROY
COLLEGE OF HUMANITIES AND SOCIAL SCIENCES,
MASSEY UNIVERSITY
PALMERSTON NORTH, NEW ZEALAND

INTRODUCTION You would not buy a car or house with faults and a limited guarantee, so why buy software like that? This question is presented as part of an argument for rejecting proprietary software and considering the use of open source software (Wyles & Clayton, 2004). Questions like this, along with similar comments in presentations concerning open source Learning Management Systems (LMS), often fail to look completely at the pros and cons of using open and malleable software. Instead, they focus the argument against the software you might be using within your institution by suggesting that you will be able to avoid the "big brother" commercial companies, might be able to circumvent your own information technology support department, or will get to tinker with the inner workings of something you might previously have been unable to touch. Is this really the way in which we should promote the benefits of open source software—by talking about what it is not rather than what it is?

Of concern is that it is incredibly easy to see open source software through rose-tinted spectacles. The open source user community is massive, vibrant, supportive, collaborative, and incredibly active—but not always. It is important when looking at open source projects to realise that the projects you don't see, the projects that failed or forked (development being split and taking separate paths) or had no clear direction, are as important to acknowledge and understand as those active projects being talked about throughout the e-learning community.

BEHIND THE CONCEPT OF OPEN SOURCE An analogy can help explain the nature of open source software. Imagine a group of parents on a working bee who come together to build a community playground. Somebody probably came up with the idea because of a perceived need, and the passion and enthusiasm they have finally gets a group of people together to start the building process. The people that join in come with a variety of skills. Some will be more suited at involving themselves in the planning stages or in project management. Others might get stuck in and be far more hands-on with hammer and nail. Others might make lunch on the sidelines, brew the cups of coffee to supply the working group, or may just make a financial contribution to the cause. The project is collaborative. It is owned by everyone and developed together. People are free to come and go as they want and free to take whatever they have learned from the experience and use it elsewhere, to build again another day. The development of open source software is based upon a similar approach. A perceived need for a piece of software then sees a group of experts band together to develop the software and make it available to anyone wanting to make use of it.

The birth date of the open source movement is somewhat ill defined, and its origins are often associated with the ongoing development of the Internet and Web technologies. The concept of open source was in fact in existence for many years prior to the Internet, but a defining moment in its history includes the attempt by the developers of the Netscape Web browser to commercialise and dominate the browser market as surfing the Web became more popular (Newman, 1999).

These attempts went against the standing ethos of freely distributable and shared programming. For many years academia had thrived on the ability to exchange technical developments with colleagues for the sake of research. Those wanting to make money from such idealistic aspirations were now challenging this right.

Richard Stallman's frustration at attempts at the commercialisation of software code, his development of the GNU license (the license by which many open source projects are distributed), and his announcement that he would develop a Unix-like operating system for all to use are other important milestones in the open source movement (Stallman, 1998). The latter event led to the involvement of Linus Torvalds who, in the early 1990s, developed the kernel for the Linux operating system, which is perhaps the best-known open source project.

Recently the use of open source software has gained prominence thanks to the Internet and the ability to communicate and establish widespread online communities, as well as the ability to quickly distribute software. At the same time, an increase in the "distrust of Microsoft's dominance has pushed the previously fringe elements of free and open source software into the lime light" (Siemens, 2003).

THE ADVANTAGES OF AN OPEN SOURCE LMS The collaborative nature of open source software is something that many people take tremendous comfort in. Glass (2003) suggests, "The open source movement is analogous to a utopian society. In utopian societies, people tend to believe they are onto something so powerful that it is fundamentally life-transforming, and they are willing to devote themselves wholeheartedly to the utopian movement of their choice" (p. 23). As Reynolds (2003) points out, "These products are good because there is a collaborative community surrounding each of them in which people listen to the ideas of others and are willing to change." Many in the open source community also see the projects they are involved in as allowing for the freedom that proprietary software fails to offer.

The main developer of the Moodle Course Management System (http://www.moodle.org) places tremendous emphasis on the collaborative nature of the project and is actively involved in researching and promoting the most effective way to continue its development, promoting what he calls a "social constructionist pedagogy" towards its development (Dougiamas & Taylor, 2003). The Special Interest Group in Open-Source Software for Education in Europe (SIGOSSEE) sees the collaborative and adaptable nature of open source as advantageous. They believe that open source software allows for the reflection of particular course content or pedagogical approaches, for the fostering of a "community" of institutes that support the exchange of ideas and concepts, and is an approach that allows for learners to become involved in improving the software. They also believe many institutions make use of open source systems because of the institution's need to cut costs.

Within a New Zealand context, many extol the virtue of open source LMS, specifically because the products are free in cost and free from ownership by a commercial company. This means not only are you able to be involved in using an open source LMS—the institution implementing the tool can access its code, determine how it works, alter code, and have more freedom over how to incorporate it into its structure and existing systems. Ultimately the open source community hopes these alterations are fed back into the community for the betterment of the open source project as a whole. LMS such as WebCT and Blackboard started their life within a university environment with the freedoms noted here. While it would be interesting to consider their path of development within an open source approach, many still suggest that even open source developers "should ultimately involve commercial software providers in their efforts" (Olsen, 2003).

THE REALITY OF OPEN SOURCE

The analogy presented earlier has another side. Picture summer days of garage sales and bake-offs, but also recall the devastation caused to a project by a key parent leaving town. Remember the arguments during the project and the sheer frustration of seeing the first children rushing to use equipment being from parents who steadfastly refused to get involved or contribute. These memories portray another face of open source software development.

Approaches used for development of an open source project, the climate under which creation, bug fixing, and code writing occur, are as important to the success of a project as the functionality that is the end result. The approach of having multiple developers who choose themselves whether to stay involved and (more importantly) how to be involved means there is tremendous independence within any development group. Therefore, when considering using an open source LMS, it is important to look at the history of the project in some detail, take the time to discover just how active the project is, and determine the level of support (both from the user community and from other sources) that it receives.

What has been interesting to note, with recent enthusiasm towards open source LMS, is how little people know of the historical events surrounding the software's lifestyle. Given that it would be vital to consider these sorts of details with a major commercial purchase, the same robust approach should be taken when looking at using open source. Indeed, given that open source usually relies heavily on the volunteer developer base to the project, it is important to take some time to understand just how strong that base is. In that respect, one must set aside the enthusiasm of using such software and address the realities of development.

As an example, consider ILIAS, a popular open source LMS and one of three systems short-listed for an evaluation by the eCDF-funded Open Source VLE project. For the early part of 2004, the ILIAS Web site (http://www.ilias.de) had as its major news item that the University of Cologne had funded another nine months of development. It announced this with the proclamation, "We will go on." The fact that a substantial research project in New Zealand short-listed a piece of software proudly promoting the saving of its demise (for at least nine months) does not reflect upon the research project, but on the fragility of open source. While "open" often means public and readily visible, we should not immediately assume that a visible presence online, combined with available patches, updates, and a wider talkative community associated with a project, implies either a large and stable developer base or strong future development.

Because of the large range of roles within an open source community, it is important to realise that the idea of a "community base" within an open source context does not necessarily mean an active base of people involved in the project. Simply making use of an open source project does not mean that a person is an active contributor. So although it is easy to look at the activity occurring within an open source LMS or to look at the suggested "install base" of such a project, such figures can be misleading. Given the hands-on and freely distributable nature of open source software, such data will not distinguish between a large and active installation or someone simply downloading and trying out the software. In reality, it is easy to find examples of just how quickly an open source project's progress can be affected even though that project may appear to be in heavy use.

Given the nature of open source projects—the freedom to be involved, the freedom to determine the extent of that involvement, and the freedom to express opinion and disagree with the underlying principles of a project—it is possible for key players or participants to hinder project development easily, particularly if their contribution plays a vital role in the software's development. Because of the "openness" of open source projects, the internal politics of development are often far more public than anything we might see in the commercial software development world. The recent history of Xoops illustrates this point.

Xoops is a highly popular open source Content Management System with an activity rating on the Sourceforge site (http://sourceforge.net) of 99.86 percent (meaning it has an incredibly active community) around the time of writing. The Xoops community was thrown into turmoil in the middle of 2004 when one of the most highly revered developers (known only as Catzwolf) closed his sister site and replaced its main page with a lengthy explanation as to why he had shut his site down. Part of his statement said:

> I have come to a stage where I do not believe in the direction that Xoops is taking now and I cannot ignore this anymore. What was once in my eyes a clear path forward has become nothing but a farce and I have become disillusioned

in its purpose and I do not see this changing. (http://wfsections.xoops2.com/)

What took place over the 48 hours after this posting is as interesting to the history of the project as the software itself. With 47 postings by users to a thread in the forum on the main Xoops site, entitled "What happened to Catz?", and more than 700 users reading this thread, visitors to the site witnessed frenzied activity as people tried to establish why the events had taken place, with members of the Development Team confessing they were as much in the dark as anyone else.

Postings quickly appeared with titles such as "Putting it in perspective" and "Is Xoops development dead?" and appeared to conclude with a posting entitled "Announcement regarding Catzwolf's rash departure," where the leader of the Core Development Team explained he had just made a two-hour transatlantic phone call to Catzwolf to resolve issues. At the same time, he announced that Catzwolf was back on board and would be co-sharing core module development responsibilities, which was an area that Catzwolf's posting had criticised. It is interesting to note that some four months later, Catzwolf did indeed end his involvement (albeit quietly this time) with the project and has never returned.

Xoops is also a curious project if you delve further into its history. Xoops is based on another open source project, but was spun off in a different direction (known as "forked" in the coding world) by a group of disgruntled developers who left the first project. Xoops itself has also been forked into a project know as e-Xoops, which became embroiled in arguments in the middle of 2004 as the development team became divided over something as simple as a name change to the project. With nothing but a voluntary commitment to the project, developers came and went as half the team wanted to retain a name connecting it to Xoops, while others felt it was time to move forward and break all ties with the software from which it was initially spawned.

Although commercial software development may also suffer from disagreements over coding and usability, there is greater consideration and stability with many of these projects, as well as greater thought given to a more holistic view of the software, including documentation. Conversely, with an open source project, you are more likely to find someone admired for his or her programming brilliance as opposed to a methodical development approach (Wilson, 1999). In many cases, it is left to the community growing around a project to develop the support and documentation.

In addition, open source projects are founded on coding that may be sufficiently complex in nature to require high-level expertise. Despite the concept of open source software suggesting accessible and malleable software, "The notion of the average user feeling free to change the open source product is a highly mixed blessing, and one unlikely to be frequently exercised" (Glass, 2003). Many at the end-user level may have comments and suggestions to see the future development of an LMS enhanced, but their ideas may fall flat if not supported by the community or by a developer willing to make coding changes in order to implement the ideas. So even with an open source project, users can become reliant on those with skills superior to their own. In the case of Xoops, Catzwolf could be seen by some as a hero, while others might view him as a precocious developer who got his way through grandstanding.

What of such events by a programmer in the commercial software world? While commercial software packages do sometimes reach their "end of life," the decision to do so is taken using more business-like criteria (sales, cost of development, etc.) than on the basis of a personal grievance. Such an event, Dalziel (2003) says, often "splits the original open-source developer community into separate groups, potentially weakening both efforts" (p. 5).

WHY ARE YOU REALLY INTERESTED? Those contemplating implementing or becoming involved with an open source LMS should take the time not only to look at the history of the project, the number of developers involved, and the technical requirements of the system, but also to consider what would happen if the project were to fail. Take the time to find out just who the people are that you might be about to rely on (don't be quick to believe you'll rely upon yourself) and just how many people truly are actively involved. Not actively enthusiastic, but actively involved. In "Open Source: Beyond the Fairy Tales," Gabriel and Goldman (2002) point out that many see the use of open source development as a way of reducing costs of coding and believe the project will evolve faster as many developers will be working on it. The reality is, they say, "A typical open-source project attracts relatively few outside developers" (p. 1).

Any historical inspection of a project should also determine whether a project has received funding. However, while it might seem easy to assume that an open source e-learning project that receives funding is likely to be more successful, Dalziel (2003) paints a more realistic picture:

> Most open-source e-learning projects have not arisen spontaneously from the goodwill of freelance software developers. They are typically the result of government or foundation funding, where developers are paid for their contributions to the project (either as contractors or as salaried employees of organizations such as universities). In the wider open-source movement, a voluntary community of developers supports projects such as Apache or Linux, hence their ongoing development is independent of the vagaries of project funding. This is not the case in e-learning, making any given open-source developer community highly susceptible to collapse when project funding ends. (p. 5)

Current New Zealand open source initiatives, exploring the use of an open source Learning Management System, saw more than $1 million allocated to research projects through the e-Learning Collaborative Development Fund. It is encouraging for all interested in this area to see research and development being funded on such a scale. However, in order to move forward with open source software projects like the ones occurring in New Zealand, projects that imply free software, it is important to acknowledge that time, effort, and funds will continue to play a major role in the success in developing and using them. Philip Long, Senior Strategist for the Academic Computing Practice at MIT, summed it up when asked if one value of the open source Course Management Systems was saving money. His response was succinct and to the point: "I'm very sorry that so many of you think that's a likelihood, because you'll be sadly mistaken" (Campus Technology, p. 16).

It is the hidden costs in using open source software that people need to be most aware of. It is not only the developers who need support during the lifecycle of a project, but the user community. While some may not see the non-technical user community as those who define an open source project's structure, it will be this group that ultimately determines its success or failure. A vibrant community of users must be funded and encouraged in their use and support of a project, including the development of material to achieve the pedagogical goals behind using the chosen LMS software.

If open source products are seen as a better solution to the commercial products that many use, what is it about the functionality of a commercial product (ignoring their so-called lack of malleability) that no longer appeals? In many instances, the answer will be hard to find but begs the question as to whether those who are enthusiastic are actually looking at what the software means to the end user as opposed to the end installer. Are those using a product like WebCT or Blackboard using it to its full potential or could something more be done with it? The next best thing doesn't always mean moving away from what you've already got, but working better with what you already possess.

Finally, perhaps open source software, with its community focus, should in fact open our eyes to the value of a collaborative approach in e-learning. A discussion on the NZ Open Source VLE Project Web site was on the suggestion of a centralised support system (Help Desk) for users of Moodle. Currently a number of New Zealand's tertiary institutions make use of WebCT or Blackboard, but there have never been any strong arguments by institutions to share resources. This raises the question, "If we feel a sense of community is important, why have many of us not embraced the online community that surrounds these commercial products already?"

The most likely answer to this question is that institutions view their installation of a commercial LMS as being set up differently to others', and consider that having localised support people who know the set-up and know their own staff is a more effective way to offer support. Those aiming to move to a provided service where an open source LMS is set up externally (but still able to be altered to suit each client's needs) may find that, given the malleability of such software, the commonalities that exist between various versions of the software may in fact be few and far between, and therefore centralised support may be difficult. This point highlights the tension between a centralised service and collaborative but independent functioning. The former provides economies but constrains local creativity, the latter allows development freedom but eventually leads to fragmentation.

BUYING OFF THE SHELF Scalise (2004) comments that in educational institutions the justification for using proprietary systems is often stability. However, he points out that this double-edged sword sees a trade-off between the stability the software provides and the resulting use of what is effectively a closed system, often with diminishing innovation.

It should be noted that this lack of flexibility is also in some way a reflection of the business processes involved in development of commercial software. Chris Vento, Executive Vice President for Research and Development and Chief Technology Officer of WebCT, points out the stability and rigidity of commercial software in his Web article "Open Systems and Open Source LMS: Settling the Debate for the Benefit of Higher Education" (2004):

> Commercial Open System providers have made significant and continuous investments in building and sustaining highly scalable, extensible, and comprehensive products. Commercial Open Systems incorporate quality and performance engineering/testing, ongoing software maintenance, a formalized feature enhancement process, customer support, and professional services required to effectively support an enterprise eLearning solution.

Beckman and Wilson (2000) suggest that when a commercial project proves troublesome, the company producing it is still likely to ship the product, whereas a troublesome open source project is most likely to be abandoned. Therefore, when reviewing open source projects that are available to use, you are less likely to find one that has a wide range of technical issues, as most of these are abandoned. They argue that this fact positively skews the success rate of open source projects in comparison to commercial software.

INTEROPERABILITY: IS MEETING IN THE MIDDLE THE SOLUTION?

As the development of both commercial and open source LMS projects continues, there has been acknowledgement of the need for open standards to allow interoperability between commercial packages and other systems. Open standards such as the Sharable Content Object Reference Model (SCORM) and the Open Knowledge Initiative have benefit in that "interoperability means that users are not locked to any one software system—they can substitute one standards-compliant system for another" (Dalziel, 2003, p. 5).

With commercial companies such as WebCT committing to many of the open standards such as those developed by the IMS consortium (of which WebCT is a consortium member), perhaps our efforts with open source lie in embracing these standards and the interoperability that can be achieved even while utilising proprietary software. Dalziel (2003) supports this idea and argues that open source software "is not free of risks and is not necessarily the most cost-effective option where commercial vendors have implemented open standards and demonstrated easy interoperability" (p. 6).

It would be financially detrimental to commercial vendors to keep their systems closed and inaccessible to other systems. The environment in which an LMS is implemented may see it surrounded by a myriad of other systems such as student management software, fees payment, results processing, and enrolment mechanisms. Integration is essential and many LMS companies not only provide systems for e-learning but products for other aspects of institutional life, such as Blackboard's Portal System, Content System, and Transaction System. Many of the developers of commercial LMS are releasing details of their APIs (Application Programming Interfaces) to third-party developers, to allow their Learning Management Systems to become a more integrated component of any wider technical architecture that an institute might employ. Examples of this include Blackboard's Extend Blackboard programme (http://www.blackboard.com/addons/b2/faq.htm#Question1) and WebCT's technical references and integration documentation.

If all live up to the ideals they are promoting and the support for open standards they aspire to, the arguments for and against the use of each type of software will not be as extreme as they first were and sometimes still seem to be. Recognition and collaboration by open source and commercial developers may ultimately see a blending of projects or even easier sharing of resources and learning objects. The focus then would not be on which system is most appropriate, but on cooperation, sharing, and ultimately improvement of all learning environments that we create in the future.

REFERENCES

Beckman, P., & Wilson, G. V. (2000, June). Open source meets big iron. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(6), 30.

Campus Technology. (2004). *Syllabus2004 conference yearbook.* Retrieved December 1, 2004, from http://www.campus-technology.com/mag.asp

Dalziel, J. (2003). Open standards versus open source in e-learning: The easy answer may not be the best answer. *Educause Quarterly, 4*, 4–7. Retrieved April 10, 2004, from http://www.educause.edu/asp/doclib/abstract.asp?ID=EQM0340

Dougiamas, M., & Taylor, P. C. (2003). *Using learning communities to create an open source course management system*. Refereed paper, presented at EDMEDIA 2003. Retrieved April 12, 2004, from http://moodle.org/doc/?frame=philosophy.html

Gabriel, R. P., & Goldman, R. (2002). *Open source: Beyond the fairy tales.* Retrieved April 25, 2004, from http://opensource.mit.edu/papers/gabrielgoldman.pdf

Glass, R. L. (2003). A sociopolitical look at open source. *Software Practitioner Newsletter & Journal of Systems and Software, Communications of the ACM, 46*(11), 21–23.

Newman, N. (1999). *The origins and future of open source software.* A NetAction white paper. Santa Barbara, California: NetAction. Retrieved April 15, 2004, from http://www.netaction.org/opensrc/future/breakdown.html

Olsen, F. (2003). *Sharing the code*. Retrieved April 15, 2004, from http://chronicle.com/free/v49/i47/47a03101.htm

Reynolds, R. (2003). *It takes a village to build good software and education.* Retrieved April 12, 2004, from http://www.xplana.com/articles/archives/Building_Good_Software_Education/print

Scalise, S.G. (2004). *The future of e-learning in learning management systems.* Retrieved April 20, 2004, from http://www.syllabus.com/news_article.asp?id=8901&typeid=155

Siemens, G. (2003). *Free and open source movements, Part 1: History and philosophy.* Retrieved April 18, 2004, from http://www.elearnspace.org/Articles/open_source_part_1.htm

Special Interest Group on Open Source Software in Education in Europe (SIGOSSEE). *Web page explanation of the SIGOSSEE.* Retrieved May 16, 2004, from http://www.ossite.org/about_sigossee/

Stallman, R. (1998). *The GNU project*. Retrieved April 15, 2004, from http://www.gnu.org/gnu/thegnuproject.html

Vento, C. (2004). *Open systems and open source LMS: Settling the debate for the benefit of higher education.* Retrieved May 5, 2004, from http://www.syllabus.com/news_article.asp?id=9267&typeid=155

Wilson, G. (1999). Is the open source community setting a bad example? *Software, IEEE, 16*(1), 23–25.

Wyles, R., & Clayton, J. (2004). *Supporting free and open source solutions.* Discussion paper. Retrieved November 23, 2004, from http://eduforge.org/docman/view.php/7/40/Sustainability%20June04.pdf

---

Philip Roy *is an E-learning Facilitator for the College of Humanities and Social Sciences, Massey University, and a Flexible Learning Leader in New Zealand. The multimedia writer for NZ Macguide Magazine, he owns and operates NZMac.com. Roy uses a wide range of open source software and has contributed to many projects.*