Educational Models and Open Source: Resisting the Proprietary University

Brenton D. Faber
Clarkson University
8 Clarkson Avenue
Potsdam NY 13699
315.268.6466
faber@clarkson.edu

ABSTRACT

This paper presents an educational model derived from open source methods for computer programming. The article places this search for an alternative model within a framework of proprietary educational practices that are driven by a need for efficiency and rationalization. As an alternative model, the paper suggests that an open source derived educational process would emphasize collaborative problem based learning, working through drafts, risk taking, mentoring, user testing, releasing early and often, developing in collaboration with users, and rewarding and building from failure.

At the same time, the paper notes that such a system would have much in common with existing theories of project-based or activity-based learning and with traditional methods of research and publication in scientific endeavors. However, the paper also argues that such a method is different from the open-course or open-curriculum projects recently publicized by several well-known universities as these practices appear to emphasize derived content rather than an open representation of process, or how the content was developed.

Collaborative, problem-based learning provides constructive approaches for building corporate and community partnerships on university campuses. At the same time, the model teaches students about collaborative work practices, working as part of a larger community, and the nature of collaborative knowledge building. As such, the model reconnects knowledge creation to research communities and to communities of users and it complicates the belief that sustainable, useful innovation can occur within proprietary systems.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces,
K.3.1 [**Computer Uses in Education]:** Education, training, curriculum design, collaboration
K.4.2 [Social Issues]: Corporate interests in education; proprietary commodification of knowledge

SIGDOC'02, October 20-23, 2002, Toronto, Ontario, Canada.

General Terms

Documentation, Design, Experimentation, Theory.

Keywords

Open Source Software, Educational models, Collaborative design.

1.0 INTRODUCTION

In this paper I present and examine educational models that emerge from open source processes of software design. These models are applicable to both academic and practitioner work settings as they bring together innovation, new knowledge building, application, and implementation. Since the mid-1980's managers, theorists, and academics have described important changes in the ways individuals relate to their organizations, workplaces, and communities. These changes are evidenced in technological issues, as increased technological development and competition has created new forms and centers of power within organizations [12, 31]; global issues, as the search for new and expanding markets and inexpensive labor has grown the geographical impact of modern organizations [11, 18], and even in generational issues as new generations of workers (commonly called "gen-Xers" or "twenty-somethings") bring different assumptions about life, work, family, and community to organizations [3, 6, 21]. Together, these changes reflect the notion of modern organizations described by Lyotard [16] as centers of "pragmatic valencies" that are designed to optimize a system's performance-efficiency against social bonds of community, narrative, and heterogeneity.

Described by Lyotard as a "postmodern condition" this new system levels issues of difference, choice, or individuality by rationalizing efficiency and power towards a logic of maximum performance in any and all organizational settings. The implications of this movement towards maximum performance and efficiency can be readily seen in current debates about education. Stanley Aronowitz, for example, has criticized movements to corporatize education that focus students on career-centered curriculum and workplace-specific learning [1]. Similarly, other critics of corporate influences in education have decried the decline of liberal studies and science courses to applied occupational courses, the ethics of university administrative practices, the outsourcing of teaching positions, and the practice of various university investment schemes as evidence of the rise of a rationalist, efficiency-based culture [28]. Futher, other authors have critiqued and argued against the rise of "corporate universities" and the blurring distinction between traditional

sites of higher education and new for-profit players like the University of Phoenix and DeVry Institute of Technology [13].

What is at stake in these discussions of organizational change, cultural change, and their inevitable impact on higher education, is the relationship education may or may not build between individuals, organizations, and communities. As Geoff Sauer has noted, Lyotard's prediction also rightly claimed that with the movement towards hyper-efficiency would come market segmentation and the proprietary commodification of knowledge [23]. As management searches for ways to rationalize production and make organizations more efficient, knowledge of systems, techniques, tools, and processes becomes a competitive advantage. As such, organizational and strategic knowledge can become more valuable when it is restricted from wide dissemination, public knowleedge, and use. Such restrictions have little effect on market practices already based on economies of supply and demand. However, when transported into an academic culture of knowledge creation, education, and learning, proprietary restrictions on knowledge dissemination have significant implications for schools, communities, and individuals.

For the most part, proprietary models of knowledge development and dissemination continue to dominate North American business practices. Corporate methods of innovation, research, and development continue to be well guarded and highly protected. Copyright laws have extended the duration of copyright eleven times since 1962. In 1998 copyright was extended to 95 years or 70 years past the author's death [23, p. 217]. In addition, recent actions by the entertainment industry have shut down various internet entertainment sites and jailed computer hackers because of suspected copyright violations. Given this cultural predisposition towards proprietary knowledge, it is not surprising that a proprietary model has found so much traction in current debates about higher education. As Sauer notes, given that a four-year U.S. university degree can cost upwards of $100,000, academic knowledges are "ripe prospective territory" for commodification [23, p. 219]. In addition, the potential wealth of new patents, sponsored research, and clinical trials, leveraged against relatively inexpensive labor and captial costs, make universities appealing places for proprietary investment. As a result, the relationship between educational structures and knowledge dissemination is a key feature in the current debate over the corporatization of education.

Proponents of the proprietary university would see the knowledge created by university researchers and students withheld from broad public dissemination but provided to the highest bidder. This educational market would then "trickle-down" to the point where university departments, classes, research, and longevity would be subject to the same conditions: those able to attract the most students and sponsorships would receive the greatest rewards. Those unable to attract or retain students or grant monies would either be forced to change or leave the university. Advocates of this market-based approach argue that it brings accountability to higher education, it ensures that courses are practical and applied, and it ensures that students and employers are able to make curriculum changes as external condition dictate.

2.0 OPEN SOURCE

Against this context of expanding commercial interests in innovation, knowledge creation, education, and dissemination has evolved an alternative form of innovation and product development commonly described as "open source." There are several emergent incarnations of open source [14] including open standards projects, open course materials (such as that recently announced by MIT), partially open projects, where some forms of collaborative software code are buried behind closed systems (Macintosh), and even a new process of intellegence gathering in public spaces called "open source intellegence" [25]. As a method for developing software, open source refers to collaboratively built code that is shared by developers and users as they co-create a product. Developers are geographically dispersed, often are unfamiliar to each other, do not work for the same organization, and represent varying levels of programming experience. In a typical open source project, a developer will work from existing or partial code to build a new tool or utility. The developer will post the evolving code to a newsgroup whose members will try out the program and provide feedback. At times members will provide their own programming solutions to problems the original developer may be having or to issues the original developer had not seen. In this way the software is built collaboratively. The final product is then posted to the group and the code is made freely available. If other programmers wish to change the code, they must repost their changes to the wider community.

Initially, Linus Torvalds resisted allowing people to sell his open source product, the Linux operating system. Originally, his policy was straightforward, "you can use the operating system for free, as long as you don't sell it, and if you make any changes or improvements you must make them available to everybody in source code" [27, p. 94]. However, once Linux had established momentum and was seen as a recognized, coherent operating system, Torvalds allowed others to sell the product so long as the other conditions were still followed [27, p. 96].

Most versions of open source, such as those associated with Linux, Perl, Apache, and Mozilla, cite Eric Raymond's *The Cathedral and the Bazaar* [20] as a founding influential document for open source. Here, Raymond describes his own efforts to build fetchmail, a web-based mail program, which he undertook as a deliberate test of the open source process. Another important source for open source is the Open Source Initiative (www.opensource.org), a nonprofit corporation that manages and promotes the open source definition and certifies software as officially open source. According to this group,

> "The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And

> this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing [19].

As Erik Berglund and Michael Priestley have noted, the open source definition does not exclude the sale of open source systems. However, it is the control over the source code that is key to open source systems [2].

2.1 Open Source Development Process

Berglund and Priestley raise a key point about open source when they write that the open source certification itself "does not really describe the nature of open source development" [2, p.134]. Berglund and Priestly note that projects such as Linux, Mozilla, and Apache relied on large, dispersed, and independent groups of programmers to contribute to the software's development. It is in this development process, and in the ways these disparate groups work together to create software, that open source can be seen to provide an educational model. Raymond lists several lessons from his own development experience that can help inform the way we look at this process. The following is a partial list of Raymond's conclusions:

> 1. Every good work of software starts by scratching a developer's personal itch.
> 2.Good programmers know what to write. Great ones know what to rewrite (and reuse).
> 3. If you have the right attitude, interesting problems will find you.
> 4. When you lose interest in a program, your last duty is to hand it off to a competent successor.
> 5. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
> 6. Release early. Release often. And listen to your customers.
> 7. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
> 8. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong [20].

In addition to Raymond's conclusions, several academic studies of open source development have provided some insight into the development process by focusing on two relevant issues: motivation to innovate and coordinating the open source development process. These studies have suggested that open source developers are motivated to produce software because of the way the process builds technologically superior software, generates collective wealth (e.g. Red Hat), and provides social benefits in the forms of altruism, reputation, ideology, and enjoyment. [10, 15, 17, 25]. However, discussions of coordination in open source projects have led to conflicting interpretations of the process. Whereas some scholars have stressed the use of highly structured governance models [17], and strong leadership, which includes vision and delegation [15], others reported parallel development by loosely organized participants [10] and a lack of team work and project sharing in the development phase but extensive peer review and modification after publication [30].

3.0 OPEN SOURCE EDUCATIONAL MODELS

Emergent in Raymond's conclusions and in the findings presented above are various starting points for an educational model based on the open source development processes. In what follows, I will outline these points and suggest some of the ways they would be relevant to educational settings. Then, in the paper's conclusions, I will return to the initial cultural discussion that initiated the paper to examine the kinds of relationships and social visions such a model brings to the classroom.

3.1 Problem-Based Learning

"Every good work of software starts by scratching a developer's personal itch"

Open source is project and problem based. Developers work on projects that interest them and by working on interesting and meaningful projects they also learn correlative knowledge, skills, and aptitudes. Similarly, an educational classroom could be based around specific projects that students find interesting, motivational, and compelling. These projects could integrate experiences from allied organizations in either open-ended case formats or in actual projects. Students would choose the projects they want to work on and as they complete these projects they would also learn other course relevant material. For example, if a group of students decided to build an on-line training module for an organization, they would need to learn not only web design issues but also instructional design, implementation, organizational communication, aspects of human resources and training, working with clients, project management, presentations, reports, and the training content. However, rather than an instructor dictating that this material must be learned, the students would come to realize that in order to produce quality work they would need to learn these things. In this case, rather than being the master of course content, the instructor faciliatates the student learning and experience, coaching the students with suggestions and options, leading them to resources, texts, and relevant information, and providing other advice to help them complete their projects.

3.2 Working from Texts, Working through Drafts

"Good programmers know what to write. Great ones know what to rewrite (and reuse)."

Experienced writers know the importance of multiple drafts and revisions. They also know that outside of a university environment, few writers ever start with a blank screen. Yet, university projects and assignments continue to perpetuate single writers working on solitary, new projects. The open source model changes this by introducing students to projects in mid-stream. Plus, it forces students to acknowledge and work with existing material, ideas, and attempted solutions; situations that are much more realistic than the perpetual "new" essay on Walden. An open source classroom would present students with half-solved problems, texts in rough draft, meeting notes, failed solutions, and dead ends. From these loose collections of texts, data, and ideas, they would forge their own texts and solutions. They would find problems

and tasks in process, learn what has already been accomplished, create new project management tasks and responsibilities, and then differenciate a new way to work on the problem. In the same way the open source builds on and integrates existing code, student projects would build on and integrate existing work.

3.3 Encouraging Risk-Taking, Inquisitiveness, Invention

"If you have the right attitude, interesting problems will find you."

An open source classroom would reward students for risk-taking, for being inquisitive, and for trying to find new ways to solve problems. As Yamauchi et al. explain, failure is a common, expected, and anticipated part of open source development and the open source community. They argue that open source developers are "biased towards action" and that "hidden experiments and failed results" are an essential part of the open source culture [29, p. 334]. Such an environment would be a significant departure from an educational culture based on examinations, standardized testing, and other rote devices that do not enable experimentation or enable students to take positive lessons from failure and experimental actions. While an examination-intensive environment may solve assessment and discipline issues for schools, such an environment does not teach students how to approach open-ended problems, nor does it provide students with the opportunities or the contexts for risk taking and for developing an inquiring mind. Torvalds writes that the hackers working on Linux become dedicated to their projects, foregoing aspects of daily life because they love programming and because they love being part of a global collaborative project [27, p. 122]. Other writes have argued that participants in open source projects are highly motivated to work and to participate in the community [9, 15, 17, 25]. The dedication of open source participants to their projects presents a unique contrast to many educational environments where educators complain that students are not motivated to learn or to do well in school or on the job. Ironically, as an undergraduate student, Torvalds built Linux outside of the classroom on his own time and did not receive academic credit for his work until his Master's degree [27, p. 136].

3.4 Handing Off Projects and Mentoring New Students

"When you lose interest in a program, your last duty is to hand it off to a competent successor."

When projects are assigned in an educational context, they are usually determined by the length of the academic term. This way, students may initiate a project, work on it for several weeks, and then complete it for credit before the term ends. However, this practice perpetuates the idea that all projects start from scratch (see 3.2 above) and must not last longer than a 12 week term. In addition, this practice does not teach students the importance of effective documentation, note-taking, and actual collaborative work because no one ever continues the projects they initiate. As a result, students do not learn how to gracefully and appropriately hand-off projects, how to strategically break projects into stages, or how to mentor new initiates into their projects. While these are key aspects of project management in the workplace, there are few contexts for even addressing these issues within the term-to-term confines of most academic schedules.

An open source classroom would extend projects from a single term or semester and would build in tools for handing each project over to a new group. Documentation would become an essential aspect of these projects because without such records the new students would be unable to take on the project and the previous students' work would be meaningless. In addition, past students would mentor new students and perhaps even compete with other groups to ensure that their projects would be continued by competent successors. This approach would teach students to see themselves as part of a larger trajectory of work rather than as solitary instantiations of one project. In addition, it would teach them to make temporal connections in their work spaces -- connections to both the past and the future. As such, it would teach students about the interdependence of project-based work and how their own work fits within larger frameworks and communities.

3.5 User Testing

"Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging."

In an open source classroom, where projects are worked on that address real-world problems, user testing becomes a vital part of everyday work. Whereas academic essays, exams, and other stagnent projects are addressed to one audience (the professor) for a one-time purpose, open source projects are successful only if people take them on and actually use them. In this way, the skills and aptitudes that technical communicators know best: audience awareness, rhetorical purpose, functionality, contextual appropriateness, become forefront in project management. Unfortunately, as many current users of open source software are aware, current documentation, interface design, and user-help practices within the open source are not good and much software is simply unusable by a generalist audience. Yet, as Berglund and Priestley have shown [2], this situation is partly due to the lack of a working framework for open source documentation and to the lack of a strong research focus on open source documentation, interfaces, and uses within the field of technical communications. We can also hypothesize that this situation is also the result of a lack of any pressing need for integrating a user-focus in most academic contexts. In an educational environment saturated with textbooks, lectures, and standardized testing, and a new drive to administrative and educational efficiency, students have little opportunity to think about and practice user testing and user-centered design. Similarly, their own experience as "users" of these educational systems is largely devoid of any evidence of user-centered approaches.

An open source classroom would build user-testing into assignments and project managment. At the same time, the course would "practice what it preaches" in its own curriculum, administration, and approaches to students.

3.6 From Drafts to Final Product

"Release early. Release often. And listen to your customers."

A crucial feature of the open source process is the release-

feedback-suggestions-revision process. This process provides the developer with user feedback, suggestions for product improvement, and ideas and solutions from other developers. The process also creates the community of developers that is central to the open source movement. By linking individual developers to a larger programming community, open source builds important social structures for developers. Torvalds argues that one of the reasons for the success of open source is that the process enables people to find places within a social order [27, p. 246-248]. By including such a process within an open source classroom, students will learn how to work within a larger community of developers/inventors, how to collaborate with each other, and how to learn the social aspects of collaboration: how to provide useful feedback, how to take criticism, how to integate other's ideas into your own work, how to share your ideas with others.

A release early, release often context teaches students to build a community of workers and it helps to introduce them to the various opportunities and problematics of organizational life which most students will face once they graduate. Release early/release often is also directly opposed to the lessons students learn from the solitary tasks of writing independent papers that have no larger audience and receive no peer or instructor feedback prior to their final draft. Such a context only furthers the proprietary, secretive culture noted in the introduction to this paper. In addition, proponents of open source methods also argue that solitary methods produce inferior products. Although coding itself, like writing, is a solitary activity, Raymond writes that,

> . . .the really great hacks come from harnessing the attention and brainpower of entire communities. The developer who uses only his or her own brain in a closed project is going to fall behind the developer who knows how to create an open, evolutionary context in which bug-spotting and improvements get done by hundreds of people. [20]

By integrating a release-feedback-revise loop into the classroom, students will learn how to leverage the strengths of their communities to create superior projects and they will begin to see themselves not so much as lone individuals but as members of larger communities. In this way, the model teaches project-related lessons and lessons in individual responsibility, individual and group relations, and community awareness and partnering.

3.7 Collaborative Development

"The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better."

Collaborative development becomes the necessary consequence of the open source model. In this model, students will collaborate with other experts, with a broad range of users, and with other groups who may have an interest in their products. Thus, the consequences of development become more significant along with the responsibilities of development. For example, students may need to gain information from legal authorities, from subject matter experts, and potentially from 5th grade children who may be using the product. Again, as noted in 3.6, this process places the students in the center of a larger community, it forces them to examine their own roles within these communities, and it introduces them to the social aspects of communities. This model also more accurately reflects the ways students will be working once they graduate and join organizations and work teams where innovation is a shared process, where they will be working with peers from a variety of backgrounds and specialist areas, and where their success will depend on the success of their work team and their ability to create products that people will want to use.

3.8 Rewarding and Building from Failure

"Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong"

Finally, open source classrooms will need to find ways to reward failure and turn failure into a positive lesson. To do this, classrooms must allow students the time and the space to run into conceptual obstacles, to undertake directions that may not work, and to hit walls in their development. At the same time, the instructor must create a learning context that is flexible enough to show students what they have learned even though they may not have completed their project (see 3.4) or solved their problem. These are not easy challenges for curriculum development or for highly motivated students who have not been allowed to experience failure or who correlate failure with self-esteem, future success, and self-image. As Raymond notes, powerful lessons can be learned by realizing that one's approach to the problem was wrong. Often, more inventive thinking will result when students have hit the wall or when their initial ideas have proven to be unsuccessful.

3.9 Model in Summary

Taking these points together creates an educational environment in which students actively work to solve real problems in collaborative environments. These problems will be multi-layered and will extend beyond the life-cycle of a single term. They will be complex and open-ended enough to require multiple approaches, starts and stops, risk taking, significant and real user testing, model releases, and collaborations. Students will work on problems that they find personally challenging and motivating. Problems will also be complex enough to require a broad range of skills, tools, aptitudes, and lessons that student will need to learn as they work on the problem.

4.0 FEASIBILITY

The purpose of this paper has been to extract and explore an education model offered by open source software development. Although future work can address issues of feasibility, practicality, and implementation, it should be noted that this model is not altogether new or radical as some may suggest. In addition, it should also be noted that this model is quite different from "open source" models currently proposed by projects like MIT's open curriculum project.

4.1 Precedents in Open Source Education

Open source proponents like Torvalds and Raymond are quick to note that this method of organizing work is not a new

invention, but actually derives from the ways academic research has been structured for years. As Torvalds recently noted, "pretty much all of modern science and technology is founded on very similar ideals to open source" [2]. Here, Torvalds refers to the time-honored system of research collaborations and peer review by which academics assist each other with their work and ensure that disciplinary work merits sufficient quality. When examining educational theory, one could argue that open source is more accurately the technological implementation of John Dewey's principles of activity-based education. For example, Dewey argued that "the first stage of contact" with any kind of education, from children through adult, must be hands-on and experiential. For Dewey, learning is a process of discovery and enactment, and of wrestling with problems first hand [5, p.160, 167].

More recently, Roger Schank has formulated what he calls "goal based scenarios" as a way to integrate an active curriculum within schools. Prevalent in Accenture's world wide training curriculum, Schank's methods have appealed to both academic and corporate educators as they provide hands-on ways for students to learn the material while at the same time experience using their knowledge in realistic, motivating situations. As Schank notes, his learning is directed at combining what one knows with what one does through a goal based scenario: "a learning-by-doing simulation in which students pursue a goal by practicing target skills and using relevant content knowledge to achieve their goal." [24, p. 165]. In these contexts, students work through simulations and receive coaching from instructors in time for them to apply learned skills to the next event or level in the simulation. At the conclusion of the simulation, students and instructors reflect back on lessons learned throughout the experience. While open source takes the simulation one step further into real life, as an educational process, it fits well and is consistent with other practice-based, or activity-based methods of education. However, in this case, open source appears to have applied these methods to a technological context and to an educational context that is not necessarily part of an academic, school-based curriculum.

On a different point, open source methods as described above should not be confused with open-text or open curriculum movements such as the policy recently announced by MIT to place all of their teaching materials on-line. While some proponents of this movement to place course materials on-line align themselves with open source, an important distinguishing feature remains that open source is a collaborative, community-building process. In many ways, open source is not about the data but it is about building connections within an expert community that collaborates to create a better product. Simply posting syllabi on-line does not bring together peers who are intersted in learning through problem solving any more than posting documentation (or a table of contents) teaches people how to program a complex piece of software. Too often, educationalists confuse process and data. Open source is about process, posting one's syllabi on-line is still about data.

4.2 Crateware

As an alternative open source educational program, we at Clarkson University are currently building an on-line resource for university faculty to collaborate together to build on-line courses in technical communication. We call this project "Crateware" (www.Crateware.org) as our inspiration comes from the modular blue crates that are used throughout university residences as furniture, bookshelves, CD racks, and moving crates. Crateware is both a site for free course ware and it is a site that enables faculty to build and improve existing courses. Following an open source model, Crateware supports an active development community that can build on-line curriculuar tools, improve existing tools, cases, lessons, or assignments, or improve the larger site itself.

Crateware courses follow Dewey's and Schank's models of active learning. Courses are posted as cases with data students organize, interpret and then use to solve organizational problems. Students solve these cases by producing recommendations, reports, actual products, and physical presentations in class. To assist students with their tasks the Crateware site hosts a variety of on-line tutorials that teach problem solving, report writing, critical thinking, analysis, and other aptitudes students need to solve their problems successfully and produce high quality products.

Our goals in creating Crateware are twofold: (1) to build a free, high quality, leading edge site for technical communications instruction that students and faculty could use with minimal overhead and training; and (2) to build a site where people interested and committed to technical communications instruction could collaborate to build leading-edge, dynamic, and useful course materials for world-wide dissemination. While the project is still in its early stages, we hope that it provides both a useful context for learning about technical communication and a collaborative process for building technical communication course resources.

5.0 Open Source, Education, and Community Participation

To return to the discussion that initiated this paper, I would like to conclude by addressing some of the social aspects of open source education. Andrew Feenberg argues that modern democracies are currently faced with two separate paths of developement. One path identifies citizenship as the roles individuals play within segmented structures such as markets, workplaces, and administrations. The other path sees individuals as agents who can surpass these systems and are not content to be mere players within predefined structures [9]. He notes that the first path correlates with Lyotard's view of the postmodern condition which emphasizes the rationalization and efficiency of these systems. Within this perspective, Feenberg writes, "the tendency is to replace human communication wherever possible by technical or bureaucratic systems" as these sytems increase efficiency, reduce the unexpected, and lead to more rational systems. At the same time, these systems also lead to an enhancement of social power by a few in the name of these increased efficiencies [9].

The vision of the proprietary university articulated in the introduction of this paper correlates with the rationalist

perspective Feenberg articulates. Such a perspective replaces collaborative research with copyright and disclosure restrictions. It replaces collaborative teaching with standardized testing. And, it replaces experiential education, discovery, and innovation, with reproduced content, static displays, and buried technology -- answers that do not reveal their questions, solutions that do not show their methods. In this system, students are rationalized as part of a larger system and they learn that success comes from learning how to efficiently adapt and move through the system. Perhaps the greatest damage that this model inflicts on education is the way it isolates students from collaborative experiences and from the collaborative nature of knowledge creation. It teaches students that knowledge can be created and innovation can be sparked by solitary thinkers working independently from each other in mutually exclusive, secret, and restricted environments.

Unfortunately, when most academics critique this model, they confuse corporate with proprietary. In some cases, these two entities denote similar interests and influences. However, in other cases, corporate interests in education do not necessarily lead to the proprietary systems Feenberg, Aronowitz, Gee, Jarvis, and others decry [1, 11, 13]. This paper argues that an open source educational model can build relationships between corporate and academic interests in ways that can benefit both contexts. In such partnerships, corporate practitioners and academics can co-create open classroom scenarios based on organization-specific problems, common situations new employees face, or yet-to-be-solved problems an organization is facing.

The open-ended nature of open source problems nicely accomodates such partnerships and the collaborative nature of the open source process invites input from both academic and practitioner settings. For example, George Dvorak, a programmer at Sun Corporation has recently argued that ongoing open source projects can be tapped by students and educators as forums for learning about programming, sharing syllabi and approaches, and for reading support materials and accessing resources {6]. But, Dvorak's suggestion could be pushed further. Corporate partners could supply projects or problems for university courses. Following the open source education model outlined above, students could work on these problems in open source teams, potentially at several different universities at once. By working together, posting ideas, solutions, and progress, and receiving feedback from corporate sponsors and faculty, the open source education community would work through the problem and provide the corporate sponsor with a final product. Yet, this model does not need to be restricted to university-corporate partnerships. Universities could also collaborate with community groups and local agencies. Such partnerships would help to integrate the university within its local community and would teach students about community activism, local issues, and community service, lessons that are rarely explored in the proprietary university [8].

As an educational model, open source provides a unique way to provide students with meaningful and motivational educational projects. It also provides students and faculty with a key way to recover the collaborative nature of knowledge creation. Open source is as much about a process as it is about building a successful project. By rediscovering this process of collaborative problem based learning, we can rebuild much of the important social infrasturcture that has been lost to rationalist systems. At the same time, we can reintroduce students to the concept of the research community and by doing so, reintroduce ourselves to new methods of innovation, knowledge creation, and inventive problem solving.

REFERENCES

[1] Aronowitz, Stanley. *The Knowledge Factory: Dismantling the Corporate University and Creating True Higher Learning.* Boston: Beacon Press, 2000.

[2] Berglund, Erik and Priestley, Michael. "Open-Source Documentation: In Search of User-Driven, Just-in-Time Writing. *Proceedings SIGDOC 2001,* 132-141.

[3] Bradford, Lawrence J., and Raines, Claire. *Twenty Something: Managing & Motivating Today's New Work Force.* Denver: Merrill-Alexander, 1992.

[4] Business: An Open and Shut Case. (2001, May 12). *Economist 359* (8221): 67.

[5] Dewey, John. *Democracy and Education.* Ed. J. Boydston. Carbondale: Southern Illinois University Press, 1985.

[6] Dvorak, George. "Collective Education" http://www.acm.org/ubiquity/views/g_dvorak_1.html

[7] Faber, Brenton. "Gen/Ethics? Organizational Ethics and Student and Instructor Conflict in Workplace Training." *Technical Communication Quarterly 10* (3), 2001: 291-319.

[8] Faber, Brenton. *Community Action and Organizational Change: Image, Narrative, Identity.* Carbondale: Southern Illinois University Press, 2002.

[9] Feenberg, Andrew. *Critical Theory of Technology.* Oxford: Oxford University Press, 2001.

[10] Feller, J. & Fitzgerald, B. (2000). A Framework Analysis of the Open Source Software Development Paradigm. *Proceedings of the 21st international conference on information systems.* Atlanta, GA: Association for Information Systems: 58-69.

[11] Gee, James Paul, Hull, Glynda, and Lankshear, Colin. *The New Work Order: Behind the Language of the New Capitalism.* Boulder Co: Westview, 1996.

[12] Hanna, Donald E. & Associates. *Higher Education in an Era of Digital Competition: Choices and Challenges.* Madison, WI: Atwood, 2000.

[13] Jarvis, Peter. *Universities and Corporate Universities*. London: Kogan Page Limited, 2001.

[14] Johnson-Eilola, Johndan. "Open Source Basics: Defnitions, Models, and Questions." *Proceedings SIGDOC 2002*.

[15]Lerner, J., & Tirole, J. (2001). The Open Source Movement: Key Research Questions. *European Economic Review 45*, 819-26.

[16] Lyotard, Jean-Francois. *The Postmodern Condition: A Report on Knowledge* trans. Geoff Bennington and Brian Massumi. Minneapolis: University of Minnesota Press, 1979.

[17] Markus, M.L., Manville, B., and Agres, C. (2000). What Makes a Virtual Organization Work? *MIT Sloan Management Review 42* (1): 13-26.

[18] Micklethwait, John, and Wooldridge, Adrian. *A Future Perfect: The Challenge and Hidden Promise of Globalization.* New York: Crown Business, 2000.

[19] OpenSource.org. www.opensource.org

[20] Raymond, Eric S. *The Cathedral and the Bazaar. http://www.firstmonday.dk/issues/issue3_3/raymond/*

[21] Reich, Robert. *The Work of Nations.* New York: Random House, 1991.

[22] Reich, Robert. *The Future of Success.* New York: Alfred A. Knopf, 2001.

[23] Sauer, Geoffry. "Community, Courseware, and Intellectual Property Law" In Chris Werry and Miranda Mowbray (Eds) Online Communities: E-Commerce, Online Education, and Non-Profit Online Activiites (pp. 215-238). New York: Pearson, 2002.

[24] Schank, Roger, Berman, Tamara, and Macpherson, Kimberli. "Learning by Doing." *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory Volume II.* Ed. Charles Reigeluth. Mahwah, NJ: Lawrence Elrbaum Associates, 1999.

[25] Stalder, Felix, and Hirsh, Jesse. "Open Source Intelligence." www.firstmonday.dk/issues/issue7_6/stalder/index/html

[26] Thau, Richard, and Heflin, Jay (Eds.). *Generations Apart: Xers vs Boomers vs the Elderly.* Amherst NY: Prometheus Books, 1997.

[27] Torvalds, Linus, and Diamond, David. *Just for Fun: The Story of an Accidental Revolution.* New York: Harper Business, 2001.

[28] Weber, Steven. "The Political Economy of Open Source Software." BRIE Working Paper 140 June 2000. Available on-line at, http://brie.berkeley.edu/~briewww/pubs/wp/wp140.pdf

[29] White, Geoffry D. (Ed.). *Campus Inc.: Corporate Power in the Ivory Tower.* Amherst NY, Prometheus Books, 2000.

[30] Yamauchi, Yutaka, Yokazawa, Makoto, Shinohara Takeshi, and Ishida, Toru. "Collaboration with Lean Media: How Open-Source Software Succeeds. *Proceedings from ACM Conference on Computer Supported Cooperative Work* (CSCW 2000).

[31] Zuboff, Shoshana. *In the Age of the Smart Machine: The Future of Work and Power.* New York, Basic Books, 1988.