High-tech Labor and Open Source

stuart mawler
smawler@gmail.com
http://www.mawler.com
January 2007

Table of Contents

stuart mawler; smawler@gmail.com
http://www.mawler.com

Abstract

In this research paper, I explore issues surrounding the labor of open source programming. I explore the nature of open source as an alternative to traditional management / labor relations as well as an organizing principle, functioning in some ways like a labor union for many of those in the field. The increasing use of software and labor that is termed "open source" by industry and entire governments warrants a close analysis of the nature of the labor that goes into its production. I build on literature regarding high-tech workers (particularly contract workers), as well as labor literature that deals with relationships that strain the traditional Marxist management / labor dichotomy, and sociological research on the motivations of open source programmers. These sources are cast against the self-descriptions of the open source community by leading members and groups. My analysis reveals problems of compensation, elitism, and overbearing ideology. I also find that open source is significantly in flux as a community and as an industry, with deep divisions in the movement's ideological goals and an increasing corporate influence, which will change the movement over time, requiring this type of analysis to be revisited in the future. However, open source remains a powerful organizational strategy and community for programming labor.

The Main Question

What is the nature of open source labor? In some very real senses, the nature of open source removes labor from the marketplace, but does this tell the whole story? Advocates of open source see the arrangement as a solution to traditional management / labor relations, the problems of labor motivation, and the problem of high-quality software. All are resolved in the leveling of the programming playing field and the emphasis on peer-to-peer, merit-based communities. As such, does open source offer an alternative to unions as an organizing principle for programming labor? There are ways in which open source communities function like unions, but there are key factors that make the parallel incomplete. Can open source be considered a true alternative to traditional organizations?

Perhaps more importantly, who cares? Open source software is free, hence the labor is free; therefore, the labor that goes into creation of the software is inconsequential, as is the organization of the laborers who create it. However, despite the "freedom" (or openness) of the source code, open source is an increasingly important part of the global marketplace. Open source makes ideological claims that strike at the heart of key Western (or at least US-based) values of intellectual property and ownership rights. Further, these ideological claims result in products that take aim at very specific (often Microsoft-marketed) products, creating real competition between open and closed source artifacts. As the open source products gain ideological credibility, governments increasingly gravitate toward them as a means to "free" themselves from closed source products, beholden to more obviously partisan viewpoints, objectives, and interests. In an even more interesting shift, open source has been adopted by corporations as a tool to move toward more profitable product niches and strike directly at competitors through the creation of alternative standards.

Open Source Background

But what is “open source” programming? To begin at the beginning, programming is the act of writing symbolic logic in such a way that a computer can be controlled to perform some function. Programming occurs in many different languages, each with a particular specialty, like graphics rendering, accounting, or mathematical algorithms. Programmers often choose to specialize in certain languages (like C or Java) or functions (like operating systems or games). Most of the public conception of software is constructed by interactions with companies who make and market this software, like Microsoft or Nintendo. These companies consider the source code for their products to be trade secrets, forming a competitive advantage in the market. In these situations, the source code is protected by copyright laws and patent laws, much like other scientific discoveries and technological innovations. To ensure the protection of the intellectual investment, the source code is not open for viewing by those outside the company—the source is closed.

Open source exists in direct opposition to the traditional corporate model. Open source indicates that the code created to make the computer perform its tasks is open to be viewed and, most importantly, updated by any competent programmer.

When computers were first brought on the scene, there was no closed source. In fact, there was little source to be had at all. Computers did not come loaded with office productivity software or even operating systems. Corporations marketed large machines to the largest corporations and government agencies, expecting them to write software to support complex computations. In the 1950s, IBM created a community (still operating in some form today) of users who would contribute their modifications back to the community, thereby improving the lot of all users and customers.[1]

However, companies, including IBM, began to improve on the software products being packaged with their hardware. Where the hardware had been the value-added product, software moved up into the most-valued slot, becoming a growth area. The combination of investment and value changed the dynamic at all software companies, encouraging them to view software as a core asset to be protected, invested in, and sold. Software became capital alongside or even above hardware.

The notion that software is property to be owned and controlled created the open software movement, making it a movement born in ideology. The ideology specifically argues that information and knowledge should be free and that ownership is wrong.[2] However, the notion of an almost inexhaustible resource pool of programming labor that might contribute to the development of open source software was difficult to approach prior to the personal computer (PC) and the internet. The PC helped create programmers who were hobbyists rather than labor, in the traditional sense of the word, taking on whatever project appealed to them, working according to whatever schedule suited their needs, ability, psychology, or social networks. With the advent of world-wide communications networks (including those linked before the internet), the labor environment changed for both the hobbyist and the traditional laborer; collaboration became possible across human networks that were previously unimaginable in scope.

Specifically, the internet changed the open source landscape, greatly increasing speed, breadth, productivity, interconnection, cooperation. One of the best known open source projects, the Linux operating system, began in 1991, leveraging the internet as its organizing and enabling structure. Earlier projects soon shifted to take advantage of the new medium.[3]

As open source has gained traction, companies have begun to "return" to open source. The reasons a company may embrace open source are as varied as the companies involved. Some see open source as a means to achieve standardization, since other companies and individuals are required to participate. With the increased participation, the software created has greater potential to become a de facto standard or cement the reputation of an existing technology as the standard, as in the case of the Java language, which Sun is planning to take from proprietary to open source. Alternatively, some companies wish to create a standard in opposition to an existing corporate-based standard. This almost guerrilla standardization is behind much open source work like the move to take Netscape from proprietary to open as an alternative to Microsoft Internet Explorer (IE), the move to create an open source office product (OpenOffice) alternative to Microsoft Office, or the potential move by Sun to make its own version of the UNIX operating system open source.[4]

There is also a potential that corporations may see open source as a form of "outsourcing", where the programming labor is distributed worldwide, but the corporation need not pay for it. However, this is not entirely fair, since any corporation that moves into open source does not do so without cost. Companies involved in open source have programmers on staff who contribute to the appropriate communities (for example, Linux, the dominant open source operating system). Further, these companies will often have advisory boards dedicated to the appropriate open source efforts and send significant numbers of staff to open source conferences to help craft the future of particular open source efforts. Hence, money is still being spent at almost the same rate as it was before. The main advantage is that the expertise available is far broader than anything a single company could purchase. More pressing for a company is the ability to use open source to shift focus away from infrastructure software (for example, operating systems) toward more lucrative value-added software. This model takes advantage of de facto and guerrilla standardization, as well as aspects of outsourcing.

IBM offers a particularly good example of this shift. One of IBM's core products has been development environments, which are products designed to make programming simpler and easier by offering tools for typing shortcuts, testing, code check-in and check-out, etc. Every language needs a good development environment, since the completeness of a toolset encourages many programmers to use a particular language.[5] As more developers opt for a language, companies can rely in the continued existence of that language, using it as the basis for further products, with some confidence in overall market stability. IBM invested years and uncounted dollars in the creation and enhancement of their development environments, yet decided to release the basic source code into the open source domain under the name "Eclipse".[6] Once compiled, Eclipse can be used as a basic development environment without further modification. However, there are both free and commercial products available to extend and enhance Eclipse. IBM itself offers a development environment called "RAD" that requires Eclipse.[7]

In this situation, IBM has used almost all of the corporate involvement paradigms. Eclipse helps stabilize the use of the Java programming language as the de facto standard in which most business programming is done, since the Eclipse environment supports Java first and foremost (though it also supports some other languages).

Eclipse also helps undermine the attempts by Microsoft to establish dominance in the programming world through their own proprietary toolset around their own proprietary languages. Microsoft created VisualBasic (not to be confused with another, older language called "BASIC") and more recently C# (pronounced "see sharp"), the latter being a direct competitor to Java. Other companies fear that if Microsoft went unchallenged, programmers would migrate to the Microsoft tools, undercutting efforts by other companies to market and control programming tools and environments.

Further, though taking a chance on the open source environment, IBM likely hopes to gain by releasing Eclipse to the broader programming community, since the target market for the tool is the very community that will be able to extend the product in new ways, hopefully making the product more stable and feature-rich. Finally, with a stable base environment available, IBM can direct its attention toward value-added features that have more marketing potential. Since there is a worldwide community of developers with a vested interest in a stable development environment (particularly one that is consistent between the office and home), IBM is relatively assured of continued attention to the product by that community. In addition, they continue to invest their own developers in the effort, though in fewer numbers than they invest for their new product for sale as an add-on to Eclipse.

It is important to note that Eclipse is not the only example. Linux has much corporate involvement with comments in the source code signed by developers from IBM, HP, and RedHat, among others. Part of the draw toward open source contribution, for many of these companies, is the appearance of good corporate citizenship, an image which hearkens back to the ideological beginnings of open source, as the antithesis of the corporate machine. While this association may seem problematic for corporations, the current programming landscape has positioned Microsoft as the personification of corporate evil-doing, allowing other companies to define their image in opposition to Microsoft.

Ideological Divisions

The division between companies (those who endorse open source versus those who endorse Microsoft) is echoed by a division within open source. Rather than being a unified community, open source is actually divided into two basic camps. Both agree that anyone can contribute to software and that source code must be available to developers because many developers together can make better software faster. Both emphasize that group membership is primarily a function of ability, with members of the group being better programmers. The split arises from a basic ideological difference that, to outsiders, may seem like the finest of hair-splitting.

On one side is the Free Software Foundation, building GNU and EMACS, among other things.[8] The primary individual in this area is Richard Stallman, a former MIT AI lab hacker.[9] People in this camp emphasize the freedom of information and intellectual pursuits, claiming that copyright is an inherently wrong-headed approach.[10]

The other side lacks a clear label to the outside world, being defined by the Free Software Foundation, as well as themselves, as open source in the strict sense. The primary leader of this area is Eric Raymond, who is a controversial figure and lightning rod.[11] This camp focuses on pragmatism, seeing open source as a means to several ends, including better software and the ability for more people to have a piece of the action and make money. They care little, if at all, about whether information is or should be free.[12] Together, these two camps are technically known as “FLOSS”, which stands for “Free/Libre and Open Source Software”,[13] though I will continue, for simplicity, to speak of them collectively as “open source”.

Literature Survey

Within the current IT world, contingency is one of the key factors with which labor must cope, as Stephen Barley and Gideon Kunda note.[14] Contingency applies particularly well to the open source community. For Barley and Kunda, the contingency is based on the status of labor with respect to management. Consultants can be hired and fired with less paperwork and hassle than employees, consulting jobs may be hard to find, consulting companies may not provide the safety net for their professionals that they might. Open source presents much the same problem or opportunity—contributors may join or leave a community easily and quickly, in theory; paying jobs may be hard to find; and there is little safety net for open source labor.

Out of the issue of contingency comes the concepts of skill and credibility, which are used to counter the contingent aspects of consulting, according to Barley and Kunda. Consultants reduce the contingency through increased credibility, established through the acquisition of new skills. Jobs are taken specifically to acquire new skills, side projects are undertaken to acquire new skills; and consultants are brought to particular contracts based on known skill sets they can bring to a team. Similarly, open source contributors may take on projects to learn new skills and increase their participation (and likelihood of making money) through their demonstrable skills.

Importantly, open source stretches traditional Marxist dichotomies of management versus labor. Like Barley and Kunda's contractors and Deborah Fitzgerald's farmers, open source contributors have greater control of their labor situation than traditional Marxist images of labor—they already control the means of production.[15] In many senses, open source contributors can function as management, playing both sides of the dichotomy, like Fitzgerald notes for farmers. Not only do these contributors write code, they may review the code of others, take "ownership" of particular modules or entire applications, and "organize" the efforts of a team on a particular problem. Further, like a farmer, the ability of an open source contributor to make a living out of the available means of production is dependent on many layers of skill beyond programming. For an open source career to be truly lucrative, the contributor must be an entrepreneur on several fronts, likely speaking, writing, and offering expert technical assistance.

Like the farmer planning for the next season, the open source contributor needs to be able to predict what fields of endeavor need to be plowed next or simply left fallow for a later opportunity.

Important to any analysis of open source labor is view of the social aspects of the work. While there is not as much published research on this specific area of endeavor, Karim Lakhani and Robert Wolf offer insight into the motivations of open source contributors.[16] Lakhani and Wolf find that, in order to be a contributor, and a significant one, "enjoyment-based intrinsic motivation, namely how creative a person feels when working on the project, is the strongest and most pervasive driver. We also find that user need, intellectual stimulation derived from writing code, and improving programming skills are top motivators for project participation".[17]

Most important, however are the self-descriptions of the FLOSS community by its leading members. In particular, the writings of Richard Stallman, head of the Free Software Foundation, and Eric Raymond, self-appointed open source apologist, support the various foci on corporate involvement, labor arrangements, and the social image of open source.[18]

A Taxonomy of Open Source Labor

The programmers actually involved in the creation, enhancement, and maintenance of open source software fall into four categories: 1) corporate contributor; 2) open source employee; 3) free lancer; 4) hobbyist; and 5) artist. For those uninvolved with the realities of open source, the first category may be surprising. As open source rises in importance to the corporate world, those corporations with vested interest in a particular open source project contribute staff to the maintenance of that software. Laborers can be found in this category at IBM, HP, RedHat, Sun, Oracle, and almost any large software firm (with the possible exception of Microsoft). This form of labor is the most similar to traditional Marxist notions of labor, in that the laborer is an employee of a corporation with layers of management. Notably, this form of labor arrangement is also the least contingent, offering significant stability in terms of pay, benefits, etc. However, these programmers are likely elite,[19] giving them a unique position within the corporation.

As representatives of the companies for which they work, they are under scrutiny from both the other community members and their corporate management. With the wider open source community, the corporate contributor needs to prove his/her abilities and independence from the goals of his/her employing company, since the open source community operates on an "internal market in reputation".[20] In his/her own company the open source contributor is an ambassador who can make or break the corporate reputation. From both perspectives, the corporate open source contributor needs to have extremely high credibility, built on a significant base of knowledge and skill in the area to which they will be contributing. Partly to solidify the credibility of the corporate contributor and to reward the perceived value of these individuals, they are tend to be granted more liberties in terms of working hours, conditions, and arrangements.

While the second category, open source employee, might sound like an oxymoron, both the Free Software Foundation and the Mozilla Corporation are not-for-profit foundations that employ programmers.[21] The pay for these jobs, being in the not-for-profit sector, is likely lower than could be had in the for-profit private sector; however, the positions come with a moral sense of righteousness not possible with most for-profit work. Those in this category are likely to be significantly driven by ideological concerns, as compensation for the lower pay.

Free lancers, the third category, may also be identified as IT contractors, spreading the contingency of employment across several different spheres of employment. However, the free lancer earns significant income from his/her work in open source. This category of labor acquires employment from the open source market place and further contract employment, based on credibility and expertise.[22] Contacts are hard to establish in this market and tools for connecting programmers to "employers" are in flux and immature at best. Importantly, this category of labor most closely approximates the situation described by Deborah Fitzgerald, with respect to farmers; free lancers are required to possess many more skills than simply programming, in order to be able to make money.

The fourth category of labor is the hobbyist. Typically, the hobbyist is a programmer by trade, but probably in a corporate (non-open source) setting. Contributions made to open source are, following Karim Lakhani and Robert Wolf, likely made for fun, challenge, or mental exercise. The day job programming for the corporation is what pays a hobbyist's bills and open source contribution is a form of escapist activity. Adding complexity, some hobbyists may also be corporate contributors or free lancers, taking on additional programming tasks simply because they seem interesting.

Finally, there is the artist. This category is slightly unparallel with the other four and intentionally so. Artists see programming as art, above even an intellectual pursuit; they engage in programming pursuits to create end products that go beyond functional to the point of beauty. Further, the code that is written by an artist must be beautiful or "elegant". The number of other programmers who will see the actual code is miniscule and the number of non-programmers even smaller, but the artistry remains, nonetheless. Programmers in any of the other categories may consider themselves artists, though not all members of any of the other categories will consider themselves artists.

Of these categories, the corporate contributor, the open source employee, and the free lancer exist within the traditional monetary market place. The hobbyist, by definition, is not seeking remuneration for his/her efforts. The artist does not necessarily eschew compensation, particularly since an artist can also be a corporate contributor, free lancer, and/or hobbyist—just as throughout history, the fortunate artist has a wealthy patron who gives him/her significant artistic latitude.

However, these are not the only means of making a living in open source. It has been widely assumed that open source programmers use their contributions as resume builders specifically to enter the corporate market as an employee, achieving greater stability. This is certainly a path that programmers have taken, but research shows that this is not a primary motivation for open source contribution and not nearly as common as previously thought.[23] This sort of transaction is very difficult, given that many of those hiring for corporate employment are not reading open source code, unless they happen to be open source contributors themselves, which, in a circular way, makes them less likely to be hiring managers.

The most lucrative path to monetary success in open source is not in programming at all, but in tasks that spring from the programming. Open source programmers are encouraged to write documentation, provide expert consulting, provide software support, offer education, and perform training. More lucrative than these, however, are two other options: speaking / lecturing and writing.[24] These two options imply a significant level of expertise and wide credibility, relying heavily on forms of knowledge outside of programming, like marketing, public speaking, and writing, in much the same way that Deborah Fitzgerald's farmers required more than physical labor for success. Importantly though, success in areas related to programming requires a basis in actual programming. Neither Richard Stallman nor Eric Raymond would be successful speaking and writing had they not already possessed programming credibility.

The net result of all this is that open source labor exists both within and outside the traditional monetary economy. Laborers can cross the boundary in many ways, but exist in a shared organizational structure with shared norms and values.

Comparing Traditional Labor Unions & Open Source Labor

One of the primary goals of traditional unions is to reduce contingency imposed by management. Unions protect wages, health benefits, and pensions through collective bargaining with management—the voice of the collective workers is louder than the one worker's voice alone. Under typical union conditions, fair treatment for workers is achieved by an emphasis on years of service rather than any attempts to quantify performance or skill; those who have devoted more time to the collective effort are given the most reward.

The open source community appears to reduce no contingency for labor, since employment is highly variable and, in some senses, available to only the most advanced practitioners. However, stability is achieved in the market, through a breadth of adoption. With more contributors to an open source project, there is a wider user base, making an open source project more likely to become a standard (for example, SendMail or Linux). Once an open source project becomes a standard, corporations increase the systemic stability further by tailoring other products to that standard. This does not guarantee employment for any specific programmer, but it could be said to create an overall market for programmers that remains consistent and the stability of an open source project is significantly influenced by the collective action of the group.

In general, it is in collective action that the open source movement and labor unions most thoroughly coincide. First and foremost, the open source community can be considered loosely democratic, which is a value shared by most traditional unions. In fact, the open source community is the more thoroughly democratic of the two, since there is little that passes for official leadership, having eliminated the “tremendous overhead” associated with management.[25] Community members do not vote on a representative to management. They do not vote on contracts that bind them in particular situations. Rather, the group decides on a particular direction based on group consensus or the powerful attraction of a particular individual’s vision. The only binding contract is the commitment to open software, which is more like the governmental provision for collective action. In addition, many key sub-groups in the wider open source community see open source in a consciously ideological manner, discussing monopolies and the rights of workers,[26] as well as the concept of a brotherhood of programmers.[27]

Unions also take outward collective action, in the sense of strikes and boycotts, and the same is true of the open source community. One IT industry commentator writes, “Linux and open source has penetrated most technical schools, government IT shops, and technology companies. Its membership, while not officially listed, is easily in the millions of people who believe in or support their version of the concept of open source, which Linux, to them, represents. There may not be a great deal of agreement on the terms, but the group can act as a group and has the tools to coordinate that action”.[28] This commentator goes on to highlight a collective action taken by the Linux community against a piece of reporting considered harmful to the Linux project. The collective sent emails to advertisers and launched electronic attacks on the company publishing the article. The collective mobilized against a perceived threat, to ensure that Linux could continue to grow and thrive.

The main difference between traditional unions and the open source community lies in the basis for promotion and respect in the two groups. Unions, coherent with their attempt to provide stability, base much of their work on tenure, while the open source ethos is based entirely on merit. A contributor has only as much respect and credibility as their code is able to show. Based on the tone of comments made in the code, collegiality is respected in open source, which is consistent with the need to constantly protect one's own reputation.[29]

Issues with Open Source

The Oppression of Poverty

Despite the hype associated with open source, it is, itself, not free of "bugs" from a labor perspective. Naturally, the primary question is whether, by endorsing open source methods, a programmer replaces the oppression of management (real or imagined) with the oppression of poverty. In other words, how can programmers expect to be paid for their work? I have outlined several means by which programmers can and do make a living in the open source arena, but these methods all have significant drawbacks. Corporate contribution involves working in a more traditional relationship with a corporate employer. Open source employees are likely going to be paid less than for-profit employees doing exactly the same work. Free lancers face significant contingency regarding payment and an uphill battle to establish contracts in a shifting and hard-to-locate marketplace. If any of these labor types should try to supplement their income with additional, extra-programming activities, they will also need to supplement their abilities in ways that may be unsuited to their psychological makeup and take time away from the central and foundational activity of programming.

Assuming a contributor chooses sides in the open source ideological debate, part of that decision rests on interpretations of intellectual property and the ability for an individual to profit from intellectual endeavor. The free software branch of open source challenges the basic notion of copyright laws, arguing that ideas are free. Naturally, if ideas are free, the labor is likely to be compensated accordingly. It is equally unsurprising to find that not all authors feel the same. One commentator says, "The GNU people assume that because there are no material costs for software then it should be free to copied [sic] and redistributed […]", however, "Those people making software would like to make sure that even though people can make copies, because there are no material costs, that they can still get paid for their labor",[30] showing that there is not universal agreement that the worker is liberated by the free / open software movement.

However, traditional unions do not necessarily offer a good solution to the problem of compensation. Unions require expensive dues, providing a barrier to entry for many people in widely varied industries. According to the open source community, the only barrier to entry is ability (though this will, most likely, require monetary investment to acquire). Further, unions place limits on the amount of compensation labor can receive, including yearly pay raises, since they also provide for guaranteed raises. In the open source marketplace, the guarantees are removed, but so, correspondingly are the limits. Contingency is traded for theoretically unlimited possibility.

Meritocracy or Elitism?

While there is not universal agreement on the compensation for programming labor, there appears to be agreement on the community norms for recognition. Instead of being tenure-based, open source is seen as a meritocracy, where a contributor's status is based on his or her abilities. Following the thoughts of Eric Raymond, unions seek to constrain the roles of developers and management for the collective advantage of programmers, which might be argued to serve the needs of the majority of programmers, rather than the exceptional, who tend to be those working on open-source.[31] Raymond would see the result as heavy regulation, with restrictions on the individual, as well as the group, leaving few satisfied.

In the writings of both Stallman and Raymond, open-source offers an alternative resulting in developers who are happier, more productive, and inherently oriented toward collective group endorsement.[32] However, the reason the group is so much better off than corporate developers has a lot to do with the self-image as elite. Raymond highlights this image when speaking of his ideal organization for a successful open-source project. He says such an ideal arrangement will have a "project core" and a "project halo", where the core is "typically quite small", on the order of one to three developers, while the halo "often numbers in the hundreds".[33] The emphasis on small leaves little room for those considered less capable. For open source, the emphasis on "merit" creates a culture that is not necessarily welcoming to all participants—programming is an elite task undertaken by a small cadre of highly skilled knowledge workers.

Unions appear to offer a more egalitarian alternative, since they welcome all comers (with payment of dues) and reward length of service, when members follow the rules. However, the result is that rewards and punishments for outstanding or lackluster performance are less likely, since this would trump tenure-based rewards. Further, it can be argued that unions do not, in fact, welcome wide participation by labor. In many jurisdictions and types of labor, the union decides who is allowed to work and for how long; the union welcomes all comers, but only on their own, sometimes Byzantine, terms.

Just Paying the Bills

The fact is, however, that programming is not necessarily a calling for many people any more than building cars for General Motors is a calling for all the workers on the assembly line. Further, there are likely more tasks to be completed than can be accomplished by the core of elite programmers that apparently dominate open source. Finally, how interesting are the tasks to be completed? Is the construction of a corporate account management system as interesting a problem as the creation of an entire operating system?

As defined, open source is more calling than contract (particularly in the free software camp). Open source contributors would be likely to agree with the notion that "units of work [...] are political in their design",[34] assigning political ends to the corporate system owners and having well-defined notions of the economics of the software industry.[35] However, many of the corporate employees, while potentially aware of the political implications of labor, are unconcerned, preferring to devote their energies to life beyond programming. They do not consider the international collection of programmers to be "comrades" linked by sharing programs as a "fundamental act of friendship",[36] rather, it is just a job. A less overtly ideological conception of labor has some advantages. Tasks considered plebian, lacking excitement, or overtly commercial may be out of the consideration for the open source community, but still essential for the operation of business or government.

A union-based approach may regularize labor, allowing workers to view the labor as more job than calling. However, unions are hardly devoid of ideology. Unions apply member dues to political campaigns, regularly take sides on global trade issues, attack particular camps that may oppose them, and possess internal politics consistent with a representative form of governance. Simply choosing to organize as a union will not remove the ideological conflicts inherent in labor.

Conclusion

Open source is significantly in flux as a community and as an industry, as evidenced by the deep divisions in ideological goals (free versus merely open), but the changes affect all aspects of the community. The level of interest in the field appears to constantly increase, causing a rise in the number of contributors, increasing usage of the resulting tools, and more influence of the open source community on the wider world.

The market place for open source labor is particularly in flux, making consistent compensation difficult. There have been several attempts at open source marketplaces, though few have survived for any length of time. Making a living from this type of labor is possible, but highly difficult, often requiring a breadth of skills well beyond programming and not likely available to all participants. As open source continues to expand, the stability of the market for practitioners should increase, allowing more contributors to make money directly through programming in the field.

Importantly, governments and industry are increasingly turning to open source as a solution to various problems. Governments see open source as a means to remove the influence of international capital and the influence of US industry, in particular. Corporations see open source as a path to defeat rivals (Microsoft in particular) and achieve stability and standardization in software while appearing to be good global corporate citizens. In either case, the open source community, which believes itself to be charting its own course, is being influenced by the needs of government and business, which is hardly ideology-neutral. At the very least, the software is being shifted to meet the needs of corporate profits. With this shift, open source may become another avenue for exploitation of labor under the banner of "freedom".

Finally, flying the flag of meritocracy does not necessarily welcome all comers, often denigrating those in the corporate IT world,[37] but it does create an organization with a relatively high degree of cohesion. The importance of the resulting community requires that we look at it as a labor organization, if not a union, per se. Open source is capable of collective action, but there are few restrictions on the community because of its informal nature, making the collective action potentially highly disruptive, even to their own goals. Of course, this may not be much different from a traditional union demanding concessions from a failing corporation.

An old saying goes: "None of us is as strong as all of us". However, there is a sarcastic take on this saying that goes: "None of us is as stupid as all of us".[38] The bottom line is that open source is an organizing principle for labor, but it is far from a panacea. It removes some labor issues, replacing them with others, just as important. Regardless of these issues, open source must be considered a powerful organizing force not unlike a union. Governments, corporations, and individual contributors should approach it with care.

Works Cited

Barley, Stephen R. and Gideon Kunda. *Gurus, Hired Guns, and Warm Bodies: Itinerant Experts in a Knowledge Economy*. Princeton University Press, Princeton, NJ, 2005.

"Bellevue Linux Users Group". http://www.bellevuelinux.org/index.html accessed 2006/10/02-15:30et.

Chance, Tom. "The social structure of open source development". *NewsForge: The Online Newspaper for Linux and Open Source*, OSTG, Inc. February 1, 2005. Available at http://programming.newsforge.com/article.pl?sid=05/01/25/1859253&tid=25&tid=89&tid=91, accessed 2006/10/03-12:45et.

Cockburn, Cynthia. "The Material of Male Power." In *The Social Shaping of Technology*. D. MacKenzie and J. Wajcman, eds. Philadelphia: Open University Press, 1985. p. 125–146.

Enderle, Rob. "The Most Powerful Labor Union in the World: Linux?" 05/30/05 5:10 AM PT. Available at http://www.linuxinsider.com/story/43413.html, accessed 2006/10/02-15:20et.

Fitzgerald, Deborah. "Farmers Deskilled: Hybrid Corn and Farmers' Work". In *Technology and Culture*, Vol. 34, No. 2. (April 1993), p. 324-343.

"GNU - Open Source vs. Labor". January 30th, 1999. Available at http://greggman.com/headlines/2001/1999-01-30-r.htm, accessed 2006/10/03-11:50et.

Haigh, Thomas. "The Corporate Origins of Open Source". Unpublished conference paper, presented at 4S, Vancouver, November 2006, p. 4. Available online at: http://www.tomandmaria.com/tom/Writing/Slides/4S2006CorporateOriginsOfOpenSource.pdf, accessed 2006/11/10.

Harney, Stefano. "Programming Immaterial Labour". *Social Semiotics*, Volume 16, Number 1, (April 2006), pp. 75-87.

Lakhani, Karim R. & Robert G. Wolf. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects". In *Perspectives on Free and Open Source Software*. J. Feller, B. Fitzgerald, S. Hissam, & K.R. Lakhani, eds. MIT Press, 2005. Available at http://opensource.mit.edu/papers/lakhaniwolf.pdf, accessed 2006/10/03-12:50et.

Levy, Steven. *Hackers: Heroes of the Computer Revolution*. Penguin, 2001.

"Monopoly: A Brief Introduction". Unsigned. The Bellevue Linux Users Group, 20 January 2005. Available online at http://www.bellevuelinux.org/monopoly.html, accessed 2006/10/02-15:30et.

Raymond, Eric Steven. "The Cathedral and the Bazaar". Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.

Stallman, Richard, et al. "The GNU Manifesto". Free Software Foundation, 1985. Available online at http://www.gnu.org/gnu/manifesto.html, accessed 2006/10/16-20:00et.

Stallman, Richard, et al. "The GNU Project". Free Software Foundation, 1998. Available online at http://www.gnu.org/gnu/thegnuproject.html, accessed 2006/10/16-20:00et.

Stallman, Richard, et al. "Why Software Should Be Free". Free Software Foundation, 24 April 1992. Available online at http://www.gnu.org/philosophy/shouldbefree.html, accessed 2006/10/16-20:00et.

Stallman, Richard, et al. “Categories of Free and Non-Free Software”. Free Software Foundation, 1996. Available online at http://www.gnu.org/philosophy/categories.html , accessed 2006/10/16-20:00et.

Stallman, Richard, et al. “Why ‘Free Software’ is better than ‘Open Source’”. Free Software Foundation, 1998. Available online at http://www.gnu.org/philosophy/free-software-for-freedom.html, accessed 2006/10/16-20:00et.

Sterling, Bruce. “A Contrarian View of Open Source”. *O’Reilly Network*, O’Reilly Media, 08/06/2002. Available at http://www.oreillynet.com/pub/a/network/2002/08/05/sterling.html , accessed 2006/10/03-12:10et.

Ware, Lorraine Cosgrove. “CIO Research Reports: Open Source Gains Momentum”. *cio.com*. CXO Media, Inc. Dec. 3, 2002. Available at http://www2.cio.com/research/surveyreport.cfm?id=48, accessed 2006/ 10/03-12:35et.

Whalley, Peter and Stephen R. Barley. “Technical Work in the Division of Labor: Stalking the Wild Anomaly”. In *Between Craft and Science: Technical Work in U.S. Settings*. Peter Whalley and Stephen R. Barley, eds. ILR Press, Ithaca, NY, 1997.

End Notes

[1] This is a relatively well-known item among mainframe programmers, but can also be supported by some research, e.g., Haigh, Thomas. "The Corporate Origins of Open Source". Unpublished conference paper, presented at 4S, Vancouver, November 2006, p. 4. Available online at: http://www.tomandmaria.com/tom/Writing/Slides/4S2006CorporateOriginsOfOpenSource.pdf, accessed 2006/11/10.
[2] Stallman, Richard, et al. "Why Software Should Be Free". Free Software Foundation, 24 April 1992. Available online at http://www.gnu.org/philosophy/shouldbefree.html, accessed 2006/10/16-20:00et.
[3] Raymond, Eric Steven. "The Cathedral and the Bazaar". Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[4] "Guerrilla standardization" is my term.
The move to take Netscape open source was not a huge success, but it could be argued that Mozilla Firefox grew out of this effort. Firefox has gained significant marketshare on the once completely dominant IE browser.
OpenOffice is available on the web at www.openoffice.org, was once known as "Star Office", and was originally built by Sun.
Most companies have a proprietary version of UNIX. HP called theirs HPUX. The combined HP/Compaq named their new version Tru64 UNIX. IBM calls their version AIX. Sun's version is called Solaris. What sets each apart is the set of features embedded in the software.
[5] IBM dropped support for a language known as Smalltalk, dealing a final blow in a competition between Smalltalk and Java for industry dominance. Without support for a feature-rich development environment, there remained little impetus to continue writing in the language, emphasizing the need for a good environment for the long-term "health" of a language.
[6] A version of the history of Eclipse is told on the project web site: http://www.eclipse.org/org/.
[7] One of the founders of Eclipse, as listed on the project web site, was Rational Software. IBM purchased Rational Software and renamed their development environment "Rational Application Developer". To some degree, calling Rational Software a founding member of Eclipse is a ruse, considering that they were already in talks with IBM at the time.
[8] GNU is an operating system that is conceptually in the same intellectual lineage as UNIX and Linux. The name is a recursive language game, being an acronym that stands for "GNU's Not UNIX". EMACS is a development environment originally written for the language LISP. EMACS is still in use and being maintained, having many fervent partisans along the lines of another editing program known as "vi".
[9] Levy, Steven. *Hackers: Heroes of the Computer Revolution*. Penguin, 2001. The first section of the book sets up the social situation of the MIT AI Lab. The last section describes Stallman's involvement in the AI Lab and his eventual departure and formation of the Free Software Foundation.
[10] Stallman, Richard, et al. "Why Software Should Be Free". Free Software Foundation, 24 April 1992. Available online at http://www.gnu.org/philosophy/shouldbefree.html, accessed 2006/10/16-20:00et.
[11] "The Cathedral and the Bazaar" is Raymond's seminal theory piece. Bruce Sterling takes aim at Raymond in an interview that highlights the level to which Raymond is a lightning rod. See: Sterling, Bruce. "A Contrarian View of Open Source". *O'Reilly Network*, O'Reilly Media, 08/06/2002. Available at http://www.oreillynet.com/pub/a/network/2002/08/05/sterling.html, accessed 2006/10/03-12:10et.
[12] Stallman and the GNU advocates are at great pains to clarify that free software is different from open source. Comparing open source software to free software, the GNU project specifically says, "It is not exactly the same class of software: they [the open source people] accept some licenses that we [the GNU project] consider too restrictive, and there are free software licenses they [the open source people] have not accepted" (www.gnu.org, "Categories"). In anther location, Stallman puts the matter differently, saying, "The fundamental difference between the two movements is in their values, their ways of looking at the world", where the Open Source camp sees a "practical" question and the Free Software camp sees non-free software as "a social problem" (Stallman, "Why 'Free Software' is better than 'Open Source'").
[13] Lindman, p. 12.
[14] Barley and Kunda, Chapter 1, "Unlikely Rebels".
[15] Here I am referencing the work of Fitzgerald, Deboarh. "Farmers Deskilled: Hybrid Corn and Farmers' Work". In *Technology and Culture*, Vol. 34, No. 2. (April 1993), p. 324-343.

[16] Here, my source is Lakhani, Karim R. & Robert G. Wolf. “Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects”. In *Perspectives on Free and Open Source Software*. J. Feller, B. Fitzgerald, S. Hissam, & K.R. Lakhani, eds. MIT Press, 2005. Available at http://opensource.mit.edu/papers/lakhaniwolf.pdf, accessed 2006/10/03-12:50et.
[17] Lakhani & Wolf, p. 3.
[18] In particular, Richard Stallman’s writings posted to the GNU project web page at www.gnu.org and Eric Raymond’s seminal work regarding open source methods, entitled “The Cathedral and the Bazaar”.
[19] Raymond, Eric Steven. “The Cathedral and the Bazaar”. Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[20] Raymond, Eric Steven. “The Cathedral and the Bazaar”. Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[21] There are open positions advertised on the Mozilla site under “Careers”, see www.mozilla.com/en-US/about/careers.html. Stallman claims to support programmers through the Free Software Foundation in his article “Why Software Should Be Free”.
[22] There are several efforts to create markets for open source projects, including www.opensourceexperts.com.
[23] Lakhani & Wolf, p. 3.
[24] Richard Stallman and Eric Raymond certainly fit this category, but so does the founder of the Mozilla Firefox project Blake Ross, who wrote *Firefox for Dummies*.
[25] Raymond, Eric, Steven. “The Cathedral and the Bazaar”, Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writins/catheral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[26] “www.bellevuelinux.org/monopoly.html”, part of: “Bellevue Linux Users Group”. http://www.bellevuelinux.org/index.html accessed 2006/10/02-15:30et.
[27] Stallman, Richard, et al. “The GNU Manifesto”. Free Software Foundation, 1985. Available online at http://www.gnu.org/gnu/manifesto.html, accessed 2006/10/16-20:00et.
[28] Enderle, Rob. “The Most Powerful Labor Union in the World: Linux? ” 05/30/05 5:10 AM PT. Available at http://www.linuxinsider.com/story/43413.html, accessed 2006/10/02-15:20et.
[29] See my unpublished Masters Thesis, “Executable Texts: Programs as Communications Devices and Their Use in Shaping High-tech Culture”, 2006-2007.
[30] “GNU - Open Source vs. Labor”. January 30th, 1999. Available at http://greggman.com/headlines/2001/1999-01-30-r.htm, accessed 2006/10/03-11:50et.
[31] Raymond, Eric Steven. “The Cathedral and the Bazaar”. Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[32] This notion is to be found throughout Stallman’s works and in Raymo nd’s work, as well.
[33] Raymond, Eric Steven. “The Cathedral and the Bazaar”. Version 3.0. Thyrsus Enterprises, 2000. Available online at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, accessed 2006/10/17-16:15et.
[34] Cockburn, Cynthia. “The Material of Male Power.” In *The Social Shaping of Technology*. D. MacKenzie and J. Wajcman, eds. Philadelphia: Open University Press, 1985. p. 138.
[35] The Bellevue Linux Users Group is a good example of this political awareness among open source programmers. For the best example, see: “Monopoly: A Brief Introduction”. Unsigned. The Bellevue Linux Users Group, 20 January 2005. Available online at http://www.bellevuelinux.org/monopoly.html, accessed 2006/10/02-15:30et.
[36] Stallman, Richard, et al. “The GNU Manifesto”. Free Software Foundation, 1985. Available online at http://www.gnu.org/gnu/manifesto.html, accessed 2006/10/16-20:00et.
[37] Raymond is very clear in this regard in “The Cathedral and the Bazaar”.
[38] This sarcasm is taken from Despair, Inc, which creates “Demotivators” (sarcastic takes on the motivational posters used by so many companies). See www.despair.com.