1999

The Open Source Revolution: Transforming the Software Industry with Help from the Government

Mitchell L. Stoltz
Pomona College

The Open Source Revolution:

Transforming the Software Industry
with Help from Government

Senior Thesis
Submitted to
Professors Richard Worthington
and Michael Erlinger

April 30, 1999

by Mitchell L. Stoltz
Computer Science/Public Policy Analysis
Pomona College

Abstract

A new method for making software is stealthily gaining ground in the computer industry, offering a promise of better, cheaper software and the empowerment of the user. The open source movement could revolutionize the software industry...if it succeeds.

Open source means software that you are allowed to copy, modify, and give to friends. Source code, the lists of instructions which tell computers how to run, is readily available, allowing you to look inside the workings of a program and change it to suit your needs. A group of programmers, companies, users, and activists have gathered in support of this empowering technology, seeking to persuade businesses and users that open source is the way to go.

However, open source faces stiff challenges. The economic basis for the software industry is to charge users by the copy when they buy software. Copying and modification are illegal. The industry and its customers are so mired in this worldview that the idea of giving out a program's "recipe," along with a license to change or copy it at will, seems preposterous. Powerful players in the software industry, such as Microsoft, see open source as a threat to their bottom line, and have devoted their energies to discrediting and marginalizing the movement.

Beginning from the assumption that cheap, reliable software that empowers the user is a good thing, this thesis looks at the claims made by advocates about the benefits of open source. I explore how the advocates make their case to the business world, the public, and government. I also look at ways in which the government could help bring about an open source revolution, using the policy tools of procurement, research funding, standards enforcement, and antitrust law. I conclude that programmers and public interest lobbyists must join forces to carry this revolution forward, and that the time for action is now, while Microsoft is on trial.

Acknowledgments

I owe many thanks to Professor Rick Worthington for his patience, advice, and encouraging words. I also extend my gratitude to Audrie Krause and Nathan Newman of NetAction for helping to keep the open source issue alive and me in the loop, and for answering a lot of random late night questions.

Finally, my everlasting thanks to Janne, for being here and seeing me through a lot of long weekends in front of the computer. I couldn't have done it without you.

Table of Contents

List of Tables

Chapter 1
Introduction: A Software Revolution

> "The paradox of the electronic frontier is that in spite of its vast potential, we have never figured out what it means, or what it *should* mean, to buy, sell, and own goods that can be copied and transported so readily."
> -Brad Cox, Superdistribution:
> Objects as Property on the Electronic Frontier, p. 19

The high technology industries have repeatedly proven that there are few *technical* problems that can't be solved. Computers and the Internet have made advances at a dizzying rate, influencing every aspect of life, and as with all major technological advances of the past, people are finding that basic assumptions about the economic and social ordering of society are challenged by the new technology. Leaping technical hurdles is easy compared to solving the social and economic problems created by the new technology. Knowledgeable consumers begin to ask questions: Is information technology too expensive to benefit everyone in society? Has power in the software industry been concentrated into the hands of monopolists? Must we accept the inevitability of bugs in even the most advanced software? Is it really logical to pay by the copy for the collection of intangible bits that comprises a program, and if not, how will its creators be compensated for their efforts?

A group of programmers, computer enthusiasts, companies, and public interest groups has proposed an alternative business model for software development that seeks to answer these questions. Based on the belief that making the most of new technologies requires new organizational structures, new economic ideas, and even new value systems, this group has called for a radical change in the way software is produced and sold. Their system is called *open source software*[1].

The essence of open source software is that all software should be distributed with the explicit right to disassemble, modify, and redistribute the software that one has purchased, without paying royalties or other fees to the original creator. The term 'open source' refers to the fact that modifying software is next to impossible without access to its *source code*,

[1] It is also referred to as *free software* or *freeware*, see (Free Software Foundation 1998;1).

the lists of instructions in a programming language that comprise the recipe for the software. Thus, granting the right to modify is meaningless unless source code is readily available. In contrast, *proprietary software*, comprising the majority of software in use today, is distributed with a restrictive license that prohibits copying, resale, disassembly or modification. Proprietary software developers consider their source code to be a trade secret; most do not release it any more than Coca-Cola gives out its formula.

Open source software can be explained in many ways: it is a new development methodology in that it allows widely distributed and loosely organized individuals to contribute to a software product. It is an alternative business model because allowing free redistribution means that new mechanisms for compensation and profit must be created. It also defines a group of people, those who create open source software and champion it as a cause. Open source can be compared to an industrial revolution, as it represents a significant change in the structure of production. It can also be thought of as a grassroots political movement, in that it focuses on changing the opinions of individuals and building support from the ground up.

To those of us who were born and raised adhering to the traditional method of software development and the market structure that distributes it, open source seems like a radical idea. What incentive will programmers have to create software that will be given away for free? The idea seems almost ludicrous from within the proprietary worldview. Yet it seems to work—a handful of companies have made significant profits under the open source system, producing very reliable, professional quality software[2]. Loosely organized groups of programmers, spread around the world, have used this method of organization to produce important and widely used software, including some of the programs that make the Internet work.

---

[2] The most well known are Netscape Communications, which in 1998 released its popular Web browser as open-source, IBM, which now sells technical support for the open-source Apache Web server, and several companies which market the open-source operating system *Linux*.

Open source supporters claim their method produces better software and provides a competitive advantage. Many of them consider open source to be more equitable, to give more choice and autonomy to the consumer, and to reward producers in a fairer way, more in proportion to their efforts. They also consider open source a beneficial social innovation, as it could lower the price of computer systems for underprivileged groups in society. Supporters belong to three different groups: the international community of programming enthusiasts who call themselves *hackers*[3], those companies that have adopted the open source business model, and a small group of nonprofit public interest organizations that deal with issues of technological equity. All of these groups publicize open source through the media, raising awareness of the issue. In the past year, major U.S. and British newspapers have printed a total of 71 articles on open source, with even more appearing in computer industry magazines[4].

The idea of open source has created controversy in the software industry, as it asks companies to give up what they see as their primary means of profit—the licensing of their intellectual property. The success of open source to date has also created conflict among its supporters, over how to market the idea, and more fundamentally, *why* to market the idea: for its economic advantage or for its social benefits.

Thesis Roadmap

From a policy point of view, open source can be seen as an attempt at social and economic change. Based on the significant early successes of the idea, and the controversy it has ignited, it seems inevitable that open source will have some long-term impact on the software industry. A study of the idea, the changes it may cause, and how these changes

---

[3] Though the press often uses the word *hacker* to refer to someone who breaks into computer systems maliciously, this is a distortion of its original meaning of 'clever programmer.' Members of the hacker community refer to the malicious sort as *crackers.*

[4] This statistic comes from a search of the Lexis-Nexis Company News database on the term "open source" over the period from April 23, 1998 to April 23, 1999.

might be achieved will shed light on the future of the software industry and the way people use computers.

A potential participant in any industry conflict is the government, which gets involved in issues that affect the public and don't seem solvable by private means. The need for government involvement in the open source issue is another point of conflict within the open source movement. In this thesis, I will look at the need for government involvement, how to bring about government action on the matter, and what that action might be.

In summary, this thesis will address the following questions:

1. What changes would widespread adoption of the open source idea cause, economically, politically, and socially? Who would benefit? What are the potential downsides?
2. Which groups could most effectively initiate government action to bring about these changes?
3. Which policy instruments would be most effective?

The first half of this paper will look at open source from a variety of viewpoints, to serve as a basis for my analysis of government options, and additionally to synthesize different opinions about open source and the details of its current situation to create a starting point for future research on the issue. The first viewpoint is a legal one, looking at software licenses, where the "openness" of software is legally codified in terms of intellectual property rights. These licenses can be seen as occupying points on a scale of "openness," with purely proprietary software on one end and strictly open source on the other. Open source is not a black-and-white issue, as this section will show; openness comes in degrees. Closely related to intellectual property is a second issue of vital importance to the open source movement: standards, and who controls them. Open standards and open source go hand in hand, and policy that affects one naturally affects the other. Thirdly, the recent emergence of open source as a public phenomenon is seen by many as a reaction to Microsoft's monopoly in the software industry.

All of these factors, intellectual property, standards, and monopoly, interrelate in various ways: Strong intellectual property protection through software licenses allows control over the standards by which software products interact, which can allow for the creation of a monopoly. Open source licenses, on the other hand, strengthen adherence to open standards and prevent monopolies from forming. The information in this section comes mainly from open source license agreements and their explanatory materials, and from some essays on standards, monopoly, and copyright. I use basic economics in this section to frame the debate and explain the theories of monopoly formation which are relevant to the case, but this thesis is not meant to be an in-depth economic analysis of open source and monopoly. Hopefully, however, it will encourage others to conduct such an analysis.

Another focus of this paper, comprising the latter half of Chapter 2, is on the open source business model and the problems it claims to solve. Large proprietary software products suffer from a crisis of complexity in which the inefficiencies caused by adding additional programmers to a project cancel the added productivity of those programmers. Open source avoids this conundrum by tapping the resources of the Internet community in identifying and fixing flaws. At the level of individuals, it is based on voluntarism as well as on a desire for profit. For companies, alternative means of profit exist, such as selling technical support for the open source software they create. The result is a very different sort of software business that is potentially more responsive to individual customers' needs. Open source is a more socially responsible business model, leading to greater equity between producers and consumers, capital and labor, rich and poor.

An aspect of the open source phenomenon that I intentionally overlook is its international implications. Open source developers live all over the world, especially in Europe, and foreign countries also comprise much of the market for this software. However, comparing the policy options which are in use or proposed in different countries

is beyond the scope of this paper. The focus of this thesis is how the U.S. government could address the issue.

Open source does have disadvantages, and these must be addressed in order to paint a complete picture of the phenomenon. It suffers from a general lack of confidence, owing to the belief that software created outside of formal corporate hierarchies could not be trustworthy. Additionally, most open source software is written by programmers for programmers and other computer experts, and so emphasizes flexibility and power over ease of use. Most open source software does not have the user-friendliness of graphical operating systems like Windows and Macintosh, putting it out of reach of many potential users.

The second part, Chapters 3 through 5, lay out my analysis of open source activists' political organizing and lobbying efforts, what they have done so far, and what they could yet do in terms of enlisting the aid of government. The three major activist groups involved in promoting open source are the worldwide hacker community, represented by two small groups of open source evangelists, corporations which have adopted the method, and nonprofit public interest groups, notably the groups NetAction and the Consumer Project on Technology. Information on these groups comes mainly from their websites, essays, and other publicity materials, as well as interviews with two key activists: Eric Raymond of the Open Source Initiative and Audrie Krause of NetAction. The idea I focus on in this section is that all of these groups have unique talents and motivations to bring to the cause, and any successful attempt to change public policy must include cooperation between all three groups.

The first step in any policy effort is convincing government, and the public, that a problem exists and is worthy of action. Activists use techniques of political characterization to define an abstract phenomenon as a specific problem to be solved. Some characterizations work better than others, and the most important key to success is not the popularity of an issue, but its degree of compatibility with existing policies and areas of

concern. Specific criteria for successful problem characterizations I derive from the works of public policy theorists, especially William Browne, B. Guy Peters, and Deborah Stone.

The best issue with which to associate open source, I argue in Chapter 4, is monopoly, since the Department of Justice's antitrust lawsuit against the Microsoft Corporation has a firm place in the media and in the public consciousness at this time. Other characterizations include universal access—policies to ensure access to telecommunications for historically disadvantaged groups—and the economic argument that open source makes companies more competitive and profitable. Though making these associations may help the cause, they are not as powerful as monopoly, at least in the short term.

Finally, I look at the different policy tools that could be used by the government to promote open source and encourage more companies to adopt it. Since none of these policies have been enacted yet for this particular issue, there is no data available on which one would work best in this case. To make up for this, I compare potential open source policies to past uses of these policy tools for other causes. I will focus on two in particular. Procurement, or the use of government purchasing to affect the price and availability of a good, is potentially the most powerful tool for promoting open source with a minimum of political strife. To evaluate the effectiveness of procurement, and the pitfalls that such a policy could encounter, I compare a potential open source procurement policy with the case of solar cells in the late 1970's, in which procurement was mandated to lower the cost, but the program was cut before it could be effective. The other policy tool I focus on is actually a group of closely related policies: research grants, education, and standards maintenance, all policies inspired by the government's role in the creation of the Internet. The Internet makes an interesting policy situation: a government-initiated project, privatized in stages over twenty years, which became a powerful economic force. As open source software and the Internet have a common origin, the policies used for one may work for the other.

The paper will conclude with some thoughts on where the open source movement is heading, and my recommendations for action.

The Decline and Rebirth of Open Source

Although the term 'open source software' was coined rather recently, the idea has existed for many years. It originated in university computer science departments of the 1960's, especially the Artificial Intelligence Lab at the Massachusetts Institute of Technology. The idea of distributing source code freely was a natural offshoot of standard research practice. Researchers share results. Sharing of source code was essential for collaborative research, and so taken for granted that it was not given a name during this period. Later, when the programmers of this era began to call themselves hackers, sharing of code became known as the "hacker ethic."

In the 1970's and 80's, the commercialization of software and the rise of large commercial software companies like IBM and later Microsoft led to a dichotomy between academic "computer science" and for-profit "software engineering." The commercial software makers followed a different paradigm than the researchers: instead of distributing their work as an academic would, they took their example from the publishing and music industries, where individual copies of a work are sold with the stipulation that they must not be copied. The proprietary model began to dominate the commercial software industry. A brief history of these early eras of software development is given in (Raymond 1998;1).

Since the decline of the original software-sharing communities, open source has existed mainly on the fringes of the computing world, outside of the commercial mainstream. As Microsoft, Apple, Novell, Lotus, and other companies began to dominate the newly created personal computer market, and a great deal of business computing, software that was written to be shared was relegated to niches in the industry, primarily networking software and the Unix operating system.

Since its invention in the early 1970's, Unix, in one of its many varieties, has always been an operating system of choice for hackers. Its original writers, Ken Thompson and Dennis Ritchie of AT&T's Bell Labs, gave away both the source code and troubleshooting

advice for their operating system freely because AT&T, which was at the time the U.S. telephone monopoly, was prevented by antitrust laws from selling software. The University of California at Berkeley made numerous improvements to Unix, and they too released the source code liberally, at least to other academic institutions. Unix quickly became the primary operating system of computer science researchers, and thus of open source programmers as well.

Following the breakup of AT&T in 1985, Unix too was largely commercialized and split into multiple incompatible versions by companies seeking to gain a competitive advantage by adding unique features to the operating system. Sun Microsystems, Hewlett-Packard, IBM, and AT&T itself all released their own versions of Unix. However, open versions have always been available, especially through UC Berkeley and its BSD (Berkeley Software Distribution) corporate spinoff. Although Berkeley Unix was not "open source" according to the more rigid definition in use today, since using it for commercial gain required its owners' permission, Unix has nonetheless been the only major operating system for which complete source code was consistently available. Thus, it is a natural choice for open source programmers. With access to the operating system source code, they can write software that takes advantage of all of the operating system's features without depending on the creators to explain (or withhold) the details of those features.

It was the relative openness of Unix that led former MIT researcher Richard Stallman to choose that operating system as a basis for his GNU Project. Seeking to revive the "hacker ethic" in the days of proprietary software, Stallman founded the project in 1984 to create a complete set of "free software" utilities and programming tools (it was not then called open source). The GNU Project, which is the programming arm of the Free Software Foundation, created some of the most widely used pieces of Unix software, including the

Emacs text editor, the GCC compiler[5], and the GDB debugging program. These programs were adapted to run on nearly every version of Unix, further cementing Unix as the operating system of open source programmers. The history of Unix is recounted in (McKusick, 1998) and (Hall and Barry, 1990). Stallman (1998) describes the activities of the GNU Project in more detail.

The other area in which the idea of open source flourished in the era of proprietary dominance was in the creation and development of the Internet. The Internet Engineering Task Force, which designed, and continues to design, most of the important communications standards that make the Internet possible, operates using open source methods. "The IETF supported the concept of open sources long before the Open Source movement was formed," wrote Scott Bradner, one of the leaders of the IETF. "There is an intrinsic partnership between open standards processes, open documentation, and open sources. This partnership produced the Internet and will produce additional wonders in the future" (Bradner 1998, p. 52).

Many of the programs that operate in the background of all Internet activity are open source, and are so successful as to have no significant competition. These include the bind program, which translates names like "www.pomona.edu" into numerical Internet addresses (this must occur before almost every Internet communication), and the Apache web server, which serves over 50% of all Web pages. The Internet is thus closely related to the concept of open source.

The driving force behind the re-emergence of open source into the mainstream of commercial software development was a reaction to what many people perceived as a dangerous monopoly situation. Microsoft's steadily increasing market share in operating systems, application programs, and Internet software prompted many companies to look for a way to change the rules of competition by finding a new software development and

---

[5] A compiler is a program which turns source code into *object code*, finished software which can be run on a computer. GCC compiled source code in the C language, and later its successor, C++, two of the most popular programming languages in everyday use.

marketing technique. Yoffie and Cusumano (1999) call this idea "judo strategy:" redefining the field of competition, shifting into uncontested markets, and avoiding direct competition with more powerful players. Open source presented such a solution. At the same time, seeing the threats to their software-sharing community presented by Microsoft and its interference with open standards[6], a group of hackers called the Open Source Initiative was founded to promote open source as a viable business model, rather than just a programmers' hobby.

The rallying point for the rebirth of open source was a Unix-like open source operating system called Linux. Created by a Finnish college student in the early 1990's, Linux is now a mature product with approximately 7.5 million users, according to Red Hat Software's estimate, putting it among the top five operating systems in use worldwide. A large community of developers from almost every continent, as well as several commercial companies, maintain and update Linux. Its success as an operating system for business has given respect and credibility to the open source method. Most open source development today uses Linux, and supporters believe that Linux will eventually overtake Microsoft's line of operating systems because of its superior reliability.

Open source has made some inroads into for-profit software businesses. In March of 1998, Netscape Communications, makers of a popular Internet browser software, shocked the software world by announcing that it would release the source code to its browser and begin to accept changes and improvements from the Internet community through the Mozilla.org group. This decision arguably made Netscape the first well-known, mass marketable piece of software to embrace the open source model. Other corporate participants include IBM, which offers support for the open source web server Apache, and several companies that sell Linux. Some companies participate in the open source model without selling software, such as O'Reilly and Associates, which publishes books

---

[6] This technique is explained in Chapter 2.

about open source technologies. Table One gives a partial list of companies involved in the movement.

Table One: Open Source Companies

Company Name	*Open Source Product*	*Product Type*
Apple Computer	Mac OS X Server	operating system
C2Net Software	Stronghold	web server
Caldera Systems	Linux	operating system
Cygnus Solutions	GNUPro Toolkit	compiler
IBM	Apache	web server
Netscape Communications/ Mozilla.org[7]	Mozilla	web browser
O'Reilly and Associates	Nutshell Series	how-to books
Red Hat Software	Linux	operating system
S.u.S.E.	Linux	operating system
SSC Incorporated	Linux Journal	magazine

Open source has emerged from the realm of hobbyists, researchers, and specialized applications into profitable software businesses and the public spotlight. The continuance of this trend will depend both on the commercial success of these companies and on the efforts of open source advocates in promoting their cause to businesses, the public, and government.

[7] Mozila.org was created by Netscape as an independent entity to oversee the development of an open source web browser, called Mozilla, based on Netscape technology. Netscape continues to develop its proprietary web browser, Netscape Communicator, based in part on input from the Mozilla project.

Chapter 2
What's So Good About Open Source?

> "The software industrial revolution is a paradigm shift, a change in belief as to which exemplar is 'best' for thinking about a problem."
> (Cox 1996, p. 53)

The direct goal of the Open Source movement is nothing short of a revolution in the software industry, a change in the way software is developed and the way its developers can profit. The scenario this paper will analyze is the catching on of the open source idea, much as the Internet or mass production caught on as economic and social forces. However, this is only a procedural goal. What problems would be solved by open source, and what new problems created? Each group of open source advocates defines its substantive goals in a different way. While most policy analyses start with a problem to be solved, this one starts with a solution—open source software—which has the potential to address several different problems, both economic and social. This chapter looks at how open source is defined and the problems it might solve.

Software Licenses

> The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users...When we speak of free software, we are referring to freedom, not price.
> -Richard Stallman, "Preamble to the GNU General Public License

Fundamentally, open source is an issue of intellectual property. The legal playing field for the open source debate is copyright law, the aspect of intellectual property that assigns to authors and other creators (including programmers) the exclusive rights to copy, distribute, modify, perform, and publicly display their work.

When a user purchases software, some of the developer's rights are transferred. This transferal is specified in a license agreement, typically a page of small print included in a

software package or displayed on the computer screen. The issue facing a software developer is which rights to license, and under what conditions.

Software licenses occupy various points on the scale from open to proprietary, based on how many rights are granted to the purchaser rather than reserved to the creator. The software called open source occupies the region of fewer rights reserved to the creator, but even within this group there is a great deal of variation. A good general definition of open source software is given in the Open Source Initiative's "Open Source Definition" (Open Source Initiative 1997). These terms include:

- Allowing free redistribution of the software without royalties or other fees to the author.
- Requiring that source code be distributed with the software or otherwise made available for no more than the cost of distribution.
- Allowing anyone to modify the software or derive other software from it, and to redistribute the modified software under the same terms.

Within these guidelines, many different licenses exist. Among these, the most significant difference arises over whether modifications and derivations of a program must be distributed under the original license terms. The Open Source Definition states merely that redistribution of modified works under the same terms must be *allowed*. Some licenses, such as the BSD license and the MPL[8], allow a programmer to modify the software and release the modified version under new license terms, including making it proprietary. (Perens 1998, p. 184)

In contrast, the Free Software Foundation's "GNU General Public License" (Free Software Foundation 1998;2) which covers most of that organization's products and many others as well, *requires* any redistribution to take place under the same terms. The GPL, a cleverly written document, uses copyright law for a very different purpose than the law was intended: preserving the openness of all copies and derivatives of a software program. Permission for a user to copy or modify software is granted only if they promise to apply the GPL to all copies and derivatives. The license states, "you must cause any work that

---

[8] The license agreements for Berkeley Unix and Netscape's Mozilla web browser, respectively.

you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties *under the terms of this License*." The FSF's name for this clause is "copyleft," and it prevents any future versions of an open-source program from being "captured" by a company and redistributed as proprietary. It also prevents any code licensed under the GPL to be included in another software project, unless that entire project also has the GPL applied to it. Thus, no GPL code can ever be used as part of a proprietary software project.

The alternative to open source software, which includes the vast majority of commercial software available today, is proprietary software. A proprietary license prohibits modification, copying, or redistribution without the company's permission. It ensures that only one entity—the company or individual that created the software—has the right to make changes or even see the software's internal structure. Proprietary software is based on a tradition of strong intellectual property protection which originated in publishing, and uses copyright law to prosecute illegal copying. As a typical example, the license agreement for Microsoft Internet Explorer contains the following terms:

- You may not reverse engineer, decompile, or disassemble the software product.
- You may not distribute copies of the software product to third parties.
- You may not rent, lease or lend the software product.

"End User License Agreement," (Microsoft Corporation)

Intellectual Property versus Efficient Diffusion and Standardization

> Economists view intellectual property policy as a tradeoff (isn't everything?)—a tradeoff between the goal of rewarding and thus encouraging innovators, and the goal of "efficient diffusion" embodied in marginal-cost pricing.
> (Farrell 1995, p. 368)

To proprietary software developers, strong intellectual property protection as provided by licenses like Microsoft's is "seen as necessary in creating sufficient incentives for firms to engage in innovation" (Shurmer and Lea 1995, p. 378). Conflict arises over whether too much intellectual property protection creates economic inefficiency by inhibiting the

widespread adoption of technologies and limiting their usefulness. If a software developer, or any other type of inventor or engineer, designs a product that increases efficiency, saves energy, promotes well-being, et cetera, then the value of the innovation to society increases with the number of people using it, and with the number of different ways they can use it. It is thus in society's best interest for the innovation to be distributed as widely as possible, with fewer restrictions on accepted uses. On the other hand, fewer restrictions mean that the innovator is less able to realize a profit from each use of the innovation, and if the innovator is not rewarded according to the value derived from his innovation, he or she will have no incentive to continue innovating.

Clearly, some balance between efficient diffusion and intellectual property protection is necessary, but different participants in the debate disagree as to where the optimal point is. Mainstream economic thinking favors stronger intellectual property protection rather than promoting diffusion directly, with the assumption that beneficial technology will reach its potential beneficiaries as long as its creators are given a strong monetary incentive to create. According to Farrell (1995), "many economists believe that encouraging innovation is more important *in general* than encouraging efficient diffusion, suggesting that the balance should tilt somewhat towards protection and encouragement of innovation rather than towards encouraging low prices for existing innovation." Many software companies view the licensing of intellectual property as their primary business. In the words of an IBM employee, software companies "live or die based on their innovation. So we take very seriously threats to our intellectual property rights" (Ellis 1995). Open source activists put more emphasis on efficient diffusion, for various economic and ideological reasons which I'll discuss later, and they downplay the absolute importance of intellectual property as a means of securing compensation for software developers.

The dilemma of assigning 'ownership' to software is due to its intangibility. In *Superdistribution: Objects as Property on the Electronic Frontier* (1996), Brad Cox points out that for tangible goods, the law of conservation of mass guarantees compensation for

the people at every stage of production. The example he uses is a pencil: When someone buys a pencil, the loggers in Oregon, the graphite miners in Sri Lanka, and everyone else involved in creating and assembling the components of a pencil will be compensated, because no one else can do their job. Software, on the other hand, can be copied and transported around the world almost instantaneously, and there is no technological way to track or account for such copying. Copyright law creates a legal deterrent to copying, but this law is trivial to sidestep. The physical basis for our entire economic system, the impossibility of duplicating a good, is nonexistent in the software world, and the result is a "software crisis" born of the inability of our basic economic structure to distribute software (ibid., p. 31). Cox's crisis is a problem of social and economic organization, not a lack of technology.

A closely related concept is technical standards. "Standards processes," says essayist Lewis Branscomb (1995), "attempt to minimize redundancy, waste, and transaction costs by articulating a common public approach to technology and market development." For computers, standards are vital if any two products are to interact. Nearly every possible relationship (often called *interface specifications* and *protocols*) between two pieces of software is determined by a standard. Examples are the interaction of an application program with its operating system, the format of a data file to be read on different types of computers, and the ordering of bits on a wire carrying data through the Internet. In the computer industry, widely used standards contribute to the efficient diffusion of technological innovations. If more products interoperate with each other through standards, consumers have a wider choice of products to perform a given task, and more flexibility in combining products.

Most of the important standards in the computing world are created by agreements between companies or by independent standards-setting bodies such as the Internet Engineering Task Force (IETF) or the International Standards Organization (ISO). Some are created by government, and some, like the Microsoft Windows operating system and its

associated interfaces, become *de facto* industry standards through market forces. An organization that controls standards, whether through legal or economic means, gains a degree of power over all technologies that use the standard to interoperate with other technologies. An organization's control over a particular standard is aided by intellectual property rights and secrecy.

Once again, opinions on the optimal balance vary between organizations. On the more restrictive end are those companies who have become successful through strict control over their intellectual property, including the standards their technologies define and use. One of these is Microsoft, whose Windows operating system is arguably the most powerful 'standard' in the personal computing world. Microsoft prefers that interface specifications be proprietary and not disclosed freely, and it favors the licensing of those specifications for the creation of compatible but not competing products. For example, Microsoft would license a specification allowing the creation of application programs which are compatible with Windows, but not for the creation of a competing operating system (Band 1995).

Other companies and organizations take a position in the center. Sun Microsystems, a Microsoft competitor, argues that interface specifications should not be protected by patent or copyright, and furthermore should be published, although the *implementation* or source code of programs should remain proprietary (ibid.). Unlike Microsoft, whose core business is selling operating systems for individual computers, Sun built its business selling networked computer systems, which are more dependent on standards.

Open source inherently allows for open interfaces, as it allows both interface specifications and the underlying source code (in short, the entire product) to be shared between companies and other groups. Since anyone can read the source code to a program, the interfaces and protocols by which it interacts with other software and hardware are plainly revealed. Thus, anyone can write either a compatible or a competing product without obtaining a license from the creator.

The Monopoly Problem

One of the hottest issues in the computing world today is the ongoing accusations of monopoly abuses by the Microsoft Corporation, including the Justice Department's lawsuit against that company, which alleges "a pattern of anticompetitive practices designed to thwart [web] browser competition on the merits, to deprive customers of a choice between alternative browsers, and to exclude Microsoft's Internet browser competitors" (Department of Justice 1998). The Justice Department and consumer rights organizations allege that Microsoft has used its proprietary control over de facto software standards to (illegally) leverage its operating system monopoly[9] in order to gain monopoly control in other markets. Microsoft, the world's largest producer of software for personal computers, is the most successful example of traditional proprietary software development and marketing. Because of this, the open source movement is often characterized as a confrontation with Microsoft. The Washington Post, for example, termed the movement "The Spreading Grass-Roots Threat to Microsoft" (Leibovich 1998).

The economic basis for their argument is the theory of *network externalities,* which argues in favor of more standardization and less intellectual property protection, or in other words, for the open-source end of the spectrum. *Network effects* exist when the value of a good to each consumer is higher the more people are using it. For most goods this is not the case: the number of people who buy a particular food, for instance, does not make that food any more useful to a particular customer. Network effects occur when products interact, as is often the case in high tech. For example, a telephone is more useful the more people own telephones; if only one person in the world owned a telephone, it would be worthless. When network effects are present, a technology that is more widely used than its competitors is more valuable to each user, so more customers will choose that technology over others. This, in turn, increases its value even more, leading to a positive

---

[9] In 1997, Microsoft Windows was installed on 94.1% of all personal computers, according to Nathan Newman (1997, ch. 2).

spiral of increasing market share. "Businesses train employees in one technology and are reluctant to abandon that investment in training," writes Nathan Newman (1997, ch. 3)," while the existence of a pool of people trained in that technology encourages other businesses to adopt that technology." Thus, technologies with a small initial advantage (what Farrell calls the *first-mover advantage*), tend to retain an advantage over later entrants to the market.

In the computer and telecommunications industries, network effects encompass a large number of product interactions. Instead of the single interaction between telephones and the telephone network, the computer industry depends on the interactions between applications and operating systems, between Internet clients and servers, and between software and auxiliary services such as training, to name just a few.

Many economists (and more importantly, many policymakers) believe that network effects can lead to undesirable outcomes that will not be corrected by market forces, such as an industry standardizing around an inferior product. Network effects, they argue, can be used to maintain and extend monopolies in high-tech industries. If this is so, then "even small amounts of abusive market behavior, if it gives advantage in market share, is magnified in its returns to the abuser due to network effects" (Newman, 3). Failures of the market system are called are called *externalities* by economists, hence, *network externalities.*

According to antimonopoly advocates, Microsoft has made extensive use of network effects in building its corporate empire. Its core monopoly and chief source of network leverage is the Windows operating system. NetAction, a nonprofit group that monitors Microsoft, claims that it was the company's original operating system monopoly which allowed it to extend control into other sub-industries, as predicted by the network effects theory. By "bundling" its word processor and spreadsheet programs with Windows, Microsoft was able to capture monopoly shares of those markets as well. When the Internet exploded into mass-market prominence in the early 1990's, Microsoft overcame an initial

disadvantage in the Internet browser market by leveraging its monopoly position and steady operating system profits. Microsoft's browser, Internet Explorer, overtook the former market leader, Netscape, because Microsoft gave their software away for free and eventually integrated it directly into Windows.

Using network effects to extend monopoly power from one market to another relies on strong intellectual property protection. Because only Microsoft has access to and control over the software interface through which application programs interact with Windows, it would be nearly impossible for any other company to design an operating system which could run programs designed for Windows. This ensures that no company but Microsoft can take advantage of the network effects of the Windows monopoly.

If the network effect theory is valid, then Microsoft can use its monopoly power to compete unfairly against open source projects. An internal Microsoft memo on open source (Valloppillil 1998) describes methods for doing just this. The author of the memo recognizes that open source software has a strong connection to simple, open Internet communications standards. "Linux [as an example of an OSS program] can win as long as services/protocols are commodities," he writes. The suggested counter-strategy is to "de-commoditize" these protocols, which means to add proprietary modifications in order to create incompatibilities between the true standard and the modified one, and then use monopoly power to force Windows users and developers to use the modified protocol. This would leave open source software unable to interoperate with the majority of the world's computers.

Although open source software suffers from the effects of monopoly (and from the direct competitive attention of Microsoft, according to the memo mentioned above), open source also represents a possible remedy for the monopoly problem. In the world of open-source software, at the other end of the intellectual property spectrum from Microsoft, excluding competition by manipulating standards is impossible. Since anyone can distribute Linux, for example, or write another operating system which can run Linux application

programs, open source application developers have no need to favor a particular operating system manufacturer, and the cycle of network effects does not exist. Even if Linux captured a majority of the market for operating systems, no single company would be able to erect barriers to competition. Open source has the added benefit of being self-enforcing: as long as source code is publicly available, no company can recapture a program by declaring ownership and making it proprietary.

Farrell (1995) theorizes that in industries with strong network effects, a lower level of intellectual property protection is the better economic choice. Since a small initial advantage can lead to market dominance and the exclusion of competitors, imparting such an advantage through patents and copyrights may contribute to the formation of monopolies. Open source, a business model in which most intellectual property rights are waived, prevents this.

The negative implications of network effects in the software industry, and the existence of Microsoft's monopoly power, are not universally accepted. Conservative economists, such as Liebowitz and Margolis (1995;2), do not believe that network effects contribute to monopoly. Furthermore, they argue that network effects are not evidence of market failure. "Any network externality that is 'market mediated,'" they write, "meaning that the size of the network (the number of users) influences the price of inputs to a firm, or goods and services to a consumer," creates no imbalances that cannot be solved by the equalizing actions of the free market. These economists claim that the evidence of market failure from networks consists of "anecdotes and casual characterizations of technology" rather than solid empirical evidence. Behind their theories is an ideological position of resistance to government intervention.

> In these instances, we are told, we might be better off relying on the government, in its wisdom, to pick for us the products that will provide us the greatest value. Al Gore, for example, as the current administration's leader on matters of technology, might be relied upon to have a clearer vision of the course of technological change than would private-market actors such as Bill Gates.
>
> (Liebowitz and Margolis 1995;2)

Microsoft, for its part, vehemently denies that it holds monopoly power or stifles competition, and it cites a number of economists and politicians in its press releases who agree. "I think that the government would have a very hard time trying to make a case that Microsoft is actually charging a monopoly price for its products," said Liebowitz in a Microsoft press release. Microsoft calls on its supporters to defend its "freedom to innovate" unhindered by lawsuits or regulation (Microsoft Corporation 1998).

Network effects, and their application to Microsoft, are clearly not a universally accepted theory. However, for the purposes of this study, what is important is that this theory is accepted by and guides the actions of key government officials, pro-consumer activist groups like NetAction, and open source developers, the groups I will consider as possible policy initiators.

Intellectual property rights, efficient distribution of technology, open standards, and monopoly all affect one another. Figure One, in the Appendix, sums up these relationships.

The Software Crisis and How Open Source Solves It

> Efficiency is thus not a goal in itself. It is not something we want for its own sake, but rather because it helps us attain more of the things we value. (Stone 1997, p. 61)

Another issue that the open source movement claims to address is an efficiency issue. Open source enthusiasts claim that their method produces higher-quality software for a given investment of money and programmer time. If this is true, the widespread use of open source software would benefit both individuals, in terms of lower cost, and the economy as a whole, in terms of less waste of resources.

As background to the efficiency argument, I return to the "software crisis" described by Brad Cox. "The software crisis," he says," is not about the majority of programs, since the majority of programs are small. There is no shortage of small software, but of individuals capable of writing more...large programs are rare, expensive, and far more difficult to

produce than the small ones" (Cox 1996, p. 80). Cox compares the creation of large software products (in proprietary software companies) to a plumber who must mine ore, refine and mold metals to make pipes, as well as assembling them in a home. (ibid., p. 52) In other words, large software projects do not have access to a good collection of smaller software components which could be combined for more functionality, like assembling pieces of pipe, but must instead recreate all of these subcomponents for each project.

The reason for this lack of subcomponents is the intangibility problem mentioned earlier: it is difficult to charge software users per copy if copies can be made and distributed instantaneously. The most common solution to this problem within the paradigm of proprietary software is to attempt to tie software to tangible media, which is much harder to duplicate than abstract bits.

> Microsoft, for example, has demonstrated complete mastery of one such solution: attaching their electronic property to paper, cellophane, and plastic [the box, disks, manuals, and paper license agreements accompanying store-bought software]. This simple expedient allows their goods to be bought and sold exactly like cornflakes and detergent.
> (Cox 1996, p. 32)

Other solutions exist as well, such as hardware keys which must be attached to a computer for software to run, or passwords obtained from the vendor. The problem with these methods is that they only work for large-scale software. Requiring shrink-wrap packaging, hardware keys, or other such means for each of the hundreds of subroutines or component parts that make up a typical large program would be impractical. Thus, creators of small program subcomponents have no means of ensuring compensation for their work, and no incentive to produce such components for others. Creators of large programs must rewrite these subcomponents for each project. At best, they may reuse them within a single company, but there is no practical means of selling them to others.

Large software projects, says Cox, are the only projects that have commercial value under the proprietary system. However, large projects suffer from a crisis of complexity. Open source evangelist Eric Raymond describes these projects as being "built like

cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation" (Raymond 1997, ch. 1). Frederick Brooks, a former IBM engineer whose 1975 book *The Mythical Man-Month* is still highly regarded as a series of observations about the field of software engineering, identifies a paradox in large projects. On one hand, he claims, only small teams of programmers, "the small sharp team, which by consensus shouldn't exceed 10 people," are truly effective at producing any autonomous software component. This is because "the sheer number of minds to be coordinated affects the cost of the effort, for a major part of the cost is communication and correcting the ill effects of miscommunication" (Brooks 1975, pp. 30-31). The added cost of communication, he observes, quickly becomes greater than the added productivity of an additional programmer.

On the other hand, groups of only ten programmers cannot reasonably be expected to produce a very large piece of software in a reasonable time frame. Production cycles in the proprietary software industry are fanatically rapid, with new versions often released every six to nine months, and each version of a large program can contain millions of lines of code, far too much for a small team to handle.[10] "For efficiency and conceptual integrity," Brooks writes, "one prefers a few good minds doing design and construction. Yet for large systems one wants a way to bring considerable manpower to bear, so that the product can make a timely appearance. How can these two needs be reconciled?" (ibid., p. 31).

The conclusion of these authors is the existence of a fundamental limitation in the proprietary software model. Only large, complex software can easily be sold for profit, but large software faces exponential complexity leading either to very slow turnaround times or dramatic sacrifices in quality. While this claim is contestable and difficult to measure, there is a general feeling among computer users that software has too many "bugs," and that

---

[10] Mozilla, the open source version of the Netscape web browser, has 1.5 million lines of code. A complete distribution of the linux operating system, with all its auxiliary utilities, has about 10 million lines, and the forthcoming Microsoft Windows 2000 release is reported by Raymond to have as many as 60 million lines.

software users are forced to accept a much lower level of quality than is expected from other manufactured goods. A joke found on the Internet points out this dual standard:

> *If Microsoft Built Cars*
> • New seats would require everyone to be the same size.
> • The oil, alternator, gas, and engine warning lights would be replaced by a single "General Protection Car Fault" warning light.
> • You would constantly be pressured to upgrade your car.
> • Occasionally, your car would just die for no reason, and you'd have to restart it. For some strange reason, you would just accept this as normal.
> • Every time the lines on the road were repainted, you'd have to buy a new car.
> (Anonymous, 1999)

Even the military is content to install software it knows to be flawed and error-prone. In September 1997, the Navy missile cruiser USS Yorktown, the site of an attempt to save on labor costs by computerizing many ship functions, went dead in the water for over two hours when the Windows NT operating system that was controlling the ship's propulsion attempted to divide by zero and crashed (Slabodkin 1998).

All of the authors mentioned in this section have proposed solutions to the dilemma. Each attacks different parts of the problem, and some are more radical than others. Brooks (1975, p. 32) proposes a partial solution within the confines of the proprietary model. This solution, which he attributes to IBM engineer Harlan Mills, is to treat software engineering like a surgical team: the head programmer, like a surgeon, has complete responsibility for designing and writing all code. A staff of assistants handles administrative matters, maintaining the computer system, testing code, editing documentation, et cetera. This model reduces the complexity of a programming task. The communication overhead between the "surgeon" programmers of, say, five ten-person programming teams is much less than what would occur among fifty individual programmers. However, the underlying problem still exists, and this solution does not address the inability to market small, reusable software components.

Cox's (1996) innovative solution, which he calls "superdistribution," involves having users pay software developers for each time they *use* a piece of software, rather than each

time it is copied. The makers of software at all levels of complexity, from simple subroutines to application programs and operating systems, would add a line of code to their programs, saying in effect, "bill the user one tenth of a cent (or some other price) each time this code is invoked." A complete, large-scale application program, such as a word processor, would contain many such statements, one from each subcomponent from which the program is built. These billing requests would be stored in a secure "accounting chip" on a user's system and periodically sent to a bank or other trusted institution, which would debit the user's account. This solves the crisis of complexity on the economic front by allowing small-component vendors to receive compensation for their work based on the number of times it is used. Software could be freely distributed (in compiled, object code form only) as widely as possible, with no restrictions on distribution.

Open source advocates would undoubtedly object to Cox's solution on ideological grounds. Cox's "superdistribution" keeps most power and flexibility in the hands of the software producer rather than the user, and it is antithetical to the distribution of source code, which open source advocates see as vital. While it may solve the economic problem of the software crisis, Cox's solution causes no beneficial social change such as some open source advocates desire.

Open source software is perhaps the most radical solution to the software crisis, as it does not fundamentally address the question of how software makers are to be compensated. Open source starts with the notion that software and the source code from which it is created should be distributed for free. Advocates insist there is profit to be made using their model, but the specifics of profit are not their primary concern[11].

Open source development projects are organized very differently from proprietary projects. An open source project, explains Eric Raymond, often starts by "scratching a developer's personal itch," that is, a programmer sets out to solve a problem she finds

---

[11] Many open source advocates are concerned with increased economic efficiency, more consumer choice, and access for the poor. Generating the profits necessary to sustain these benefits is a lesser concern.

interesting. (Raymond 1997, ch. 2) As the programmer progresses, she makes the source code available, generally over the Internet. Other users, often those with some programming knowledge of their own, download and use the software. As flaws appear (bugs, security holes, et cetera), users, often motivated by an interest in the subject and a desire to participate, send in reports of the problem or fix it themselves and return the fix to the original author. They may also add new features to the program and return these to the author. In a large project, like the Linux operating system, different people take responsibility for different sections. Development projects are loosely centralized, generally with a few people writing new code and a large group of people using and actively reviewing it. The Internet is developers' primary mode of communication, allowing wide geographic distribution of participants. For example, the Linux *kernel*, or central operating program, was written collaboratively by developers on three continents.

Membership in an open source project is open to anyone with interest and some requisite degree of ability. The result is a kind of "organized chaos," or in Raymond's words, "a great babbling bazaar of differing agendas and approaches" (Raymond 1997, ch. 1). His fundamental observation about the development of Linux, and open source software in general, is that while programming is inefficient above a certain number of participants, debugging and editing of code is not. On the contrary, the more people use and test a piece of software, the more bugs will be found, and this effect is intensified if the users are programmers and can fix the bugs themselves. This is because while debugging requires communication between debuggers and the head programmer or coordinator, it does not require any significant communication between debuggers. Open source developer Paul Vixie points out that the best way to find bugs in software is to subject it to real-world use by as many users as possible, each with their own methods and idiosyncrasies of use that may bring bugs to light. (Vixie 1998) A large number of participants allows a wide diversity of approach, increasing the chances that someone will find a bug and that someone will readily be able to propose a solution. These two need not be the same person.

Distributing the debugging process avoids the crisis of complexity in the debugging step, one of the most resource-consuming phases of software engineering. In Raymond's words, "Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone. Or, less formally, 'Given enough eyeballs, all bugs are shallow.'" (Raymond 1997, ch. 4) Vixie calls this debugging process "the best system-level testing in the industry" (Vixie 1998).

Several factors complicate this basic model of an open source project. Participants may be individual programming enthusiasts or members of corporations; the motivations of each will be discussed below. Additionally, the distinction between a coordinator and a debugger or fixer as described above is arbitrary in many cases. Participants contribute varying amounts of effort, from finding bugs to writing large sections of code. Status and decision-making authority in the group are defined mainly by an individual's level of participation. In different projects, the central code-writing group may be of different size and may assume more or less of the code-writing and other tasks of the project.

Authority over which code to include in the software is also handled differently between projects. In the Mozilla web browser project, the core group of programmers consist mainly of engineers at Netscape Corporation, from which the Mozilla code originated. Being the authority figures for both Mozilla and Netscape's proprietary browser versions, they decide which contributions of code and new features will be included, and their decision is undoubtedly based in part on Netscape's business objectives. For the Linux operating system, final authority over changes is exercised by the original creator, Linus Torvalds, known as the "benevolent dictator" of the Linux project. As a final example, the Apache web server's central authority is a self-selecting committee known as the Apache Group, which has formal rules of consensus and voting to determine what code is to be included (Apache Group 1999). Figure Two, in the Appendix, illustrates an open source project.

One explanation for why open source development is more efficient is that it is based on voluntary participation, drawing on programmers' sense of enjoyment and accomplishment. These factors are generally not considered in the standard economic definition of efficiency. "Often, people derive happiness from doing or experiencing something, rather than from the value they obtain in an exchange," writes Deborah Stone. "A society conceived only as a network of exchanges fails to capture what are perhaps the most important sources of human happiness and well-being" (Stone 1997, p. 75). Non-economic motivations are especially apparent among hackers, who founded the open source movement. Hackers are motivated by a desire to solve interesting problems and to learn from the effort. Many see themselves as members of a close-knit community; writing code and sharing it with others are the primary means of participating in that community.

There is an economic explanation for the willingness of hackers to contribute to an open source project, even if there is no money involved. The currency of the hackers, according to Raymond, is reputation, and reputation is achieved by contributing code, and other volunteer work for an open source project.

> The "utility function" Linux hackers are maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers. (One may call their motivation "altruistic", but this ignores the fact that altruism is itself a form of ego satisfaction for the altruist)....[The culture utilizes] "egoboo" (the enhancement of one's reputation among other fans) as the basic drive behind volunteer activity.
> (Raymond 1997, ch. 10)

Raymond suggests that hackers derive more ego satisfaction from contributing to a larger software project, with more participants, and it is this fact that keeps software projects from splitting into multiple incompatible versions. Another pseudo-economic explanation for hackers' participation is that while open source software is free, hackers "pay" for use of the software by finding and fixing bugs as they use it, and returning those corrections to the community.

Raymond acknowledges that programmers have to eat, and that a majority of hackers have proprietary software jobs or other sources of income that allow them time to enjoy

their hobby of hacking. Raymond himself derives income from consulting and lecturing on open source. Other hackers write and sell textbooks on the software they create, or are involved in the other business aspects of open source, which I discuss below.

While the success of open source depends in part on tapping hackers' desire to contribute, that is not the whole picture. Corporations which adopt the open source strategy are faced with the same question with which this paper began: how are they to realize a profit? Who would pay for software that can be given away for free? The primary means of profit for open-source companies lies in selling support for their products. Technical support, including phone hotlines, help with installation and configuration, the writing of manuals, and training, is a lucrative business that often brings in more revenue than the software purchase itself. Software, in a sense, can serve as advertisement for the technical support to be coupled with it.

Along with technical support, companies can build intangible assets by creating a trusted brand name. The most successful example of this is Red Hat Software, which sells a boxed Linux distribution with an installer, manuals, extra software, and support services, all for about $50. Although it seems counterintuitive that users would pay $50 for software that can be downloaded for free on the Internet from Red Hat itself, the company prospers. Red Hat has successfully created a perception of trust and "officialness" surrounding its boxed Linux distribution, such that users feel they are acquiring software from a reliable source, rather than an ephemeral mob of programmers. This sense of security is invaluable for corporate managers and inexperienced users. Red Hat "sells" a friendly, coherent-looking interface between the chaotic open-source development process and users who need assurances of support and reliability.

Open source is often used in combination with proprietary software as a business strategy. Netscape, for example, released its web browser as open source to gain market share for its proprietary server products, which interact with the browser. Open source software can act as an advertisement for proprietary software. In addition, under most open

source licenses the company is free to use ideas and suggestions contributed to its open source project in a proprietary project.

Other means of profit exist as well. In *Release 2.1*, Esther Dyson (1998, ch. 6) lists alternate economic models for profiting from digital creations. Her work is focused on textual material such as magazine content rather than software, but some of the same principles apply. Dyson suggests that companies sell *intellectual process* rather than intellectual property, meaning the sale of technical skill and talent, rather than the product of that talent. In the case of software, this could apply to technical support as mentioned above, but additionally it could apply to writing software custom-made for a particular organization's needs, or adding specific features on request. In these situations, users pay for services rendered over time, rather than for use of the software.

An illustrative example of a company that profits from "intellectual process" is Cygnus Solutions, which makes a healthy profit by *porting*[12] the Free Software Foundation's GNU C and C++ compilers to new operating systems. What Cygnus sells is not the ported software, but their labor and expertise in porting it. Once created, the ported software is posted on the Internet and made available to everyone.

The potential benefits to the consumer from this approach are considerable. Ultimately, what customers want, what they pay for, is not software but a solution to their computing needs, a tool that can do what they need done. By paying for the expertise in customizing and adapting software to their needs, rather than for the general solution, often inadequate for particular needs, that is offered by proprietary software. Most of software's value to the customer could derive from this customization step, rather than the basic software, and paying for the service that adds the most value leads to more economic efficiency.

The long-term viability of open source as a business model has yet to be proven, but its use by several well-established companies seems to be a vote of confidence. If its supporters are correct, open source may benefit both individuals and the economy by

---

[12] *Porting* means rewriting software to run on differend computer types and operating systems.

creating better software for less initial investment, and by allowing unlimited distribution of beneficial software technology.

The Downside: Problems with Open Source

This section will describe some of the shortcomings of open source. In the short run, these are factors that keep open source out of the mainstream of commercial software. In the long run, they may be fundamental flaws in the idea. Policies to promote open source must of course take its possible ill effects into consideration.

Despite its successes in some areas, open source has not yet been accepted by most of the business world because of a lingering doubt about the quality and trustworthiness of this type of software. Although Raymond and his associates argue otherwise, the belief persists that a loosely organized group of volunteers cannot produce quality software. One of the specific concerns of the business world is that software for open source operating systems is scarce. While this is changing, with companies like WordPerfect and Corel releasing software for Linux, the number of well-known, trusted software vendors who have demonstrated some commitment to open source is relatively small. "The skeptics believe that only fools rush in to a bet-your-business relationship with [software] that is still primarily controlled and supported by its user community—no matter how skilled and committed that community is—instead of going with a brand-name vendor with a proven track record" (McNamara 1998).

Another issue is the availability of technical support. Proprietary software packages generally offer a phone line that users can call to receive technical help. Although business like Red Hat and Cygnus provide "one-stop," comprehensive technical support, this is not available for most open source software. Technical help is available for most open source by seeking out the creators of the program over the Internet, or others with an intimate knowledge of the program, and asking their advice. While this arguably leads to more knowledgeable support providers, it is not a "one-stop" source of advice, nor are the

potential providers obligated to provide advice—as with the software design itself, support happens on a volunteer basis. "No business in this country is going to wait for a 17-year-old beatnik to [answer its newsgroup post and] fix its problem," said a software consultant (ibid.).

The largest obstacle to the growth of open source is its lack of user-friendliness; open source is considered to be more difficult for the average user to install and run. In the early days of open source, software was written by hackers for hackers. The intended users of the software, being themselves programmers, valued innovation, speed, and technical creativity over ease of use or friendly user interface. In addition, the traditional focus of attention for open source developers has been "back-end" types of programs that do important work out of sight of the user, such as operating systems, networking, servers, and mail transport programs, rather than programs such as word processors with which a user interacts directly. Thus, a great variety of open source "desktop" applications has yet to be written.

"Linux is still a geeks' operating system," said a network manager, "one that takes a fair amount of knowledge to configure and maintain" (ibid.). Open source software offers a great deal of flexibility to the user, but that flexibility comes at the price of increased complexity. In a corporate setting, this complexity means that employees may require additional training. For a home user accustomed to "friendlier" graphical interfaces such as those in the Windows and Macintosh operating systems, this complexity may be especially daunting. Market pressures may force software innovators to overcome these problems, but in the meantime, open source is gaining a negative reputation among potential buyers.

Raymond claims that market demand for the service will force more companies to design friendlier user interfaces for open source programs. Specifically, he pointed out Red Hat's significant investment in user interface research (Raymond, interview). However, it remains to be seen whether this market demand will create a significant change.

Conclusions

Open source software is developed through loose collaborations over the Internet to which any interested party can contribute, and the resulting software is made available for free, along with its source code. Open source may make better use of programmer time and other resources by producing more error-free programs in less time. The fact that unlimited distribution and copying of open source programs is allowed and encouraged further increases its greater economic benefit. Policies that promote open source may be effective in counteracting monopolies based on the control of proprietary standards, since the openness of source code prevents a company from excluding competition through secrecy and copyright. Additionally, because of the low cost of obtaining copies, open source may promote more equitable access to technology for education and other uses.

The disadvantages of open source, at least in the short run, are primarily its lack of user-friendliness and a lack of public confidence in the idea. Originally written by hackers for hackers, to date open source software has seen little incentive for ease of use on the level needed by a home or small business user. In many cases, a lack of centralized technical support for a product makes businesses unwilling to trust it. Whether these problems can be overcome remains to be seen.

Chapter 3
Who Promotes Open Source?

The preceding chapter explored the potential effects of a switch to open source as the standard way of producing and distributing software. This chapter considers the question of which interest groups would be most effective at initiating action through government to achieve this goal. For the purposes of this study, I will group open source advocates into three categories: nonprofit public interest groups, the community of programming enthusiasts, or hackers, and software companies. Of these groups, I will show that public interest groups are best suited and most willing to bring the open source case to the attention of government, although the organizational, financial, and political limitations of the relevant interest groups will necessitate some cooperation between all three of the aforementioned categories.

The Hackers

The open source method evolved within the loosely organized international collaboration of programming enthusiasts, or hackers. Eric Raymond, as well as other members of the group, define hackers as a unique culture or tribe, a "continuous and self-conscious technical culture of enthusiast programmers, people who built and played with software for fun" (Raymond 1998;1). Hackers, then, are not just any computer programmer, but one who interacts with and contributes to this culture.

Although the term 'open source' has been coined only recently, the methods it describes have been developing in the hacker community for decades. The majority of open source software is written by members of this community, and it is their work, much of it undertaken for fun and challenge rather than for profit, which continually tests and refines methods for efficient open source development.

The hacker culture began in the computer science departments of a handful of universities in the 1960s and '70s, and to this day maintains some of the character of an

academic research effort. "Science," writes Chris DiBona, "is ultimately an Open Source enterprise" (DiBona, Ockman, and Stone 1998, introduction). Like many scientists, hackers believe in unlimited distribution for their ideas, in the need for reproducibility of results, and in peer review of each others' work.

Most of the original hackers drew researchers' salaries and did not depend on the commercial success of their software to make a living. Today, their sources of funding are more eclectic: some still work in universities, but many are either full-time employees at software companies or self-employed contractors doing programming work, or have other sources of income. Obviously, the need to profit from one's creations affects what kind of software is made, and what sorts of political associations hackers form.

Raymond's essay on "How To Become A Hacker" (1999) lists some norms of belief among the group that illustrate its character and motivations:

1. The world is full of fascinating problems waiting to be solved.
2. Nobody should ever have to solve a problem twice.
3. Boredom and drudgery are evil.
4. Freedom is good.
5. Attitude is no substitute for competence.

Note that point two is an exhortation to share the results of programming work, which is what defines open source. The fourth point is also especially significant, as it refers to the libertarian political beliefs that Raymond ascribes to hackers. "Hackers are naturally anti-authoritarian," he writes. "Anyone who can give you orders can stop you from solving whatever problem you're being fascinated by...So to behave like a hacker, you have to develop an instinctive hostility to censorship, secrecy, and the use of force or deception to compel responsible adults" (ibid.).

As a result of these beliefs, Raymond, and probably many other hackers, oppose almost any government involvement in the open source issue. "If someone is to get involved in fixing this problem, I don't want it to be the government," said Raymond in a phone interview, "Giving the government power to intervene in this way will create more harm than good. Consumers should make the choice, not have it made for them." It is

unlikely that the hackers, as a group, will seek any assistance through government. Many believe that open source's inherent strengths of reliability and efficiency will allow for its success in the marketplace without policy assistance. "The closed-source world," says Raymond, "cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem" (Raymond 1997, ch. 10). Of course, there must be enough programmers willing to give of their "skilled time" in order for this business model to work.

Although they avoid involvement with government, hackers are involved in public policy in the sense that they seek to spread their ideas, especially in the business world. If we define interest groups, as Hrebenar and Scott (1982) do, as groups that "make certain claims upon other groups or organizations in the society," then hackers certainly fit the bill. Two groups within the hacker community, Eric Raymond's Open Source Initiative and Richard Stallman's Free Software Foundation have lobbying, or influencing others' opinions, as a major goal. These two organizations also serve to exemplify two very different outreach methods employed by the hackers.

The Open Source Initiative was formed in February of 1998 by Eric Raymond, Bruce Perens, and other open source enthusiasts. Some of those involved, notably Tim O'Reilly, the owner of a publishing company that specializes in how-to books for open source software, had direct ties to the Silicon Valley business world. The OSI's mission is a "marketing campaign" to convince the corporate world of the benefits of open source.

Part of OSI's motivation in evangelizing the open source method as a tool for the software industry is to legitimize their methods and ideas. Programmers derive happiness and satisfaction from programming, and the widespread adoption of open source would assure that they can program and tinker, free from the restraints of copyright. Open source is what hackers do anyway; the more the idea is put into use, the more the hackers' ideals are validated.

Additionally, although Raymond believes that open source will inevitably become the dominant paradigm in software development, he acknowledges that large proprietary software companies, especially Microsoft, have the power to delay the success of open source through the manipulation of standards. This fear is another motivation for the hackers' lobbying efforts.

Targeting the news media is often a powerful method of affecting public opinion. Considering television, the Internet, and other technologies that increase the ability of the media to reach citizens, William Browne calls media lobbying "one of the most evolutionary aspects of lobbying" (Browne 1998, p. 101). The efforts of interest groups such as the Open Source Initiative are considered good news material by reporters, providing the issue has some public appeal. The Open Source Initiative set its sights on large companies (the Fortune 500) as targets of its lobbying, partially because of the prestigious and well-read newspapers and magazines that shape opinion in those companies (Raymond 1998;2). Developing these relationships with the media have paid off, at least in the short run: newspapers like the *New York Times*, the *Wall Street Journal*, the *Boston Globe,* and the London *Times*, have each run several articles which speak enthusiastically about Linux and open source, often quoting Raymond.

While the Open Source Initiative seeks to promote open source for its own sake, or for its economic and anti-monopoly benefits, others in the hacker community have different motivations. The Free Software Foundation, led by Richard Stallman, works to spread the idea that proprietary software is morally wrong, and that programmers have a moral duty to share their source code with others. Stallman describes proprietary software as a "promise not to help your neighbor."

> A cooperating community was forbidden. The rule made by the owners of proprietary software was, "If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them." The idea that the proprietary software social system...is antisocial, that it is unethical, that it is simply wrong, may come as a surprise to some readers. But what else could we say about a system based on dividing the public and keeping users helpless?
>
> (Stallman 1998)

To Stallman and his compatriots, the necessity of mutual aid and sharing, and of empowering users rather than producers, is the reason for creating open source software. They are not opposed to programmers obtaining compensation for their efforts, as long as that compensation is not secured by the use of proprietary licensing, which restricts the legal uses to which software can be put. The FSF programmers themselves receive donations from users, both individual and corporate, which sustains their activities. The reason why companies support the FSF, despite its policies that make cooperation with the business world difficult (these are described below), is that FSF software, especially the GCC compiler tools, are so useful as to be vital to many businesses. The quality and popularity of their software provides FSF with a means of income, despite its open nature.

The FSF's lobbying efforts are directed specifically towards the ideological goal of promoting software sharing. Their primary means of influence is a novel one, as it involves the software licensing procedure itself. Most of the FSF's software, and a great deal of other software, is covered by Stallman's GNU General Public License, which requires that all modifications to a program be distributed under the GPL. This clause essentially prevents proprietary software companies from using any code released under the GPL, because any program in which GPL code is included must become open source. The effect of the GPL is thus to disadvantage proprietary software makers and favor open source developers by allowing them to use GPL code, which includes the source code of Linux, the popular GCC compiler, and many other widely used programs.

Stallman and the FSF are uncompromising in their moral goals, believing that any dilution of their message of sharing, or of the GPL, for the purpose of gaining wider acceptance, is harmful to their cause. Stallman's rhetoric is often vehement. For example, he refers to proprietary software makers as "software hoarders." This extreme position has at times created tension in the hacker community. "Like anyone utterly devoted to a cause, Stallman has stirred controversy in the community he is a part of" (DiBona, Ockman, and

Stone 1998). Specifically, conflict arose between the Open Source Initiative and the FSF over OSI's outreach to the corporate world and its non-insistence on the GPL. The very term, "open source," was coined to distance the movement from Stallman's extremism.

> It seemed clear to us in retrospect that the term "free software" had done our movement tremendous damage over the years...Most of it came from...the strong association of the term "free software" with hostility to intellectual property rights, communism, and other ideas hardly likely to endear themselves to a [corporate] manager...FSF's actual position didn't matter. Only the fact that its evangelism had backfired...actually mattered.
> (Raymond 1998;2)

In return, Stallman (1998) accused Raymond and his associates of "setting aside the spirit of principle...to appeal instead to executives and business users, many of whom held an ideology that places profit above freedom, above community, above principle."

Because of its libertarian beliefs, the hacker community in general will not be the group to initiate any government policy on behalf of open source. On the other hand, hackers are the driving force behind the open source movement, so that any policy solution must take them into account and not alienate them.

Software Companies

Another important force in the open source issue is the companies that have embraced the idea, using open source as a business strategy through sales of support and expertise. For companies like Red Hat, IBM, Netscape, Cygnus, and others, forging a partnership with hackers in designing a business model around open source has proven effective.

Although companies can be powerful forces in public policy, their political positions are overwhelmingly dominated by the drive for profit. The bottom line, for a corporation, is what counts. Companies thus have very little leeway for taking moral stands on issues, or even expressing opinions on matters of market regulation, if such a choice does not lead to higher profits. "Any idealization of the market," says Browne, "takes a back seat unless that view merges enough with daily reality to secure profits" (Browne 1998, p. 33). Jamie Zawinski, the former head of the Mozilla project, pointed out that "for a publicly-traded

company, if a CEO makes a decision because it's the *right thing* rather than because it's the most profitable thing for the shareholders, he will lose his job, and possibly be sued into oblivion" (Zawinski 1999).

Any political support for open source by companies will thus occur only to the degree that such support benefits the company. This will definitely be the case for those companies that have staked their future to the open source movement, most notably Cygnus and Red Hat, and to a lesser degree Netscape. However, these companies necessarily lack the *ideological* commitment to open source, especially to the degree of Stallman and the FSF. For example, companies will combine open source with proprietary projects, or use one to sell the other, techniques which Stallman sees as violating the spirit of sharing and cooperation. Cygnus Solutions found itself disqualifying "managers who could not accept creating a closed-source component to our business. Open source was a business strategy, not a philosophy, and we did not want to hire managers who were not flexible enough...to meet overall company objectives" (Tiemann 1998).

Browne dismisses the myth that companies are completely adverse to regulation, for the simple reason that regulation can also help them out. It is not the presence of regulation, he says, but the degree to which it favors a company, that determines the company's lobbying efforts. Of course, taking a stand on particular government policies requires a greater amount of involvement in the process.

> As scholars have long noted, corporations do far more than just fight government regulation, which often, in the face of media and public attention, is a futile act. Businesses have learned reluctantly for years to accept regulation and work to make its inevitable presence as favorable to corporate ledgers as possible.
> (Browne 1998, p. 34)

Browne points out that corporate lobbying is not new; rather, it has gone on for most of this century, ever since the government abandoned its *laissez-faire* approach to businesses and began regulating their activities extensively. He estimates that as of the early 1980's, one-third of all lobbyists in Washington represented corporations (ibid., pp. 35-36).

The high-tech industries, on the other hand, have only recently begun this sort of involvement in public policy. Being a new industry, software existed without regulation for its first decades. As the Internet exploded into popular culture, and Silicon Valley became a major force in the U.S. economy, policymakers inevitably took notice, and the early confrontations between the software industry and government were not positive. Restrictions on the export of encryption software, once of interest only to spies, have in recent years been made the subject of bitter debate between the software industry and the federal government, with software makers saying the regulations hurt their competitiveness in foreign markets for this valuable technology. Another unpopular regulation was the 1996 Communications Decency Act, an ill-conceived anti-pornography statute for the Internet, which was eventually struck down by the Supreme Court as unconstitutional censorship.

These controversial policies led the computer industry to believe that it could no longer remain aloof from the political game. "We can't afford as an industry to step back and say, 'Ah, politics, we're purists. We only do great software,'" said a Silicon Valley executive (Gruenwald 1998). As a result, campaign contributions by software firms in 1997 and 1998 have doubled, to $5.4 million, since the last midterm election cycle in 1994. As Browne predicted, software firms including Microsoft and Yahoo, a web search engine, are opening offices in Washington to monitor policies which may affect them.

In spite of this trend for more policy involvement, one additional factor prevents companies from commitment to a single issue. In the high-tech industries, new product innovations are born, and the fortunes of companies rise and fall, in a matter of months at a time. In Washington, on the other hand, generating policy from an initial idea to the passage of laws and regulations can take much longer. A company may not be willing to commit resources for promoting a specific policy, only to have it tabled or stuck in committee, when their need for the policy may be out of date in a matter of months due to changes in technology.

Software companies are clearly more involved in the policy process then they were just four years ago. Some of them have developed expertise in policy matters which could be helpful to the open source effort, and their Washington offices are certainly good sources of policy information. Large corporate interests with demonstrated economic power are good allies in the policy process, as their voices are heeded by a pro-business Congress. However, companies are not the group to lead a pro-open-source policy initiative. As explained above, the changing realities of business makes long-term commitment to a specific policy goal impossible for a company; if open source no longer generates profit for a company, it will have no reason to support the effort. If, however, open source becomes the dominant paradigm in the industry, companies will lobby to protect it.

Public Interest Groups

The most likely organizations to initiate public policy action to promote open source may be nonprofit public interest groups committed to open source for its social benefits. Currently, the only groups of this sort which have taken an interest in open source are NetAction and its partner organization, the Ralph Nader-affiliated Consumer Project on Technology. Groups like these are distinct from organizations such as the Open Source Initiative, which is also incorporated as a nonprofit, in that NetAction seeks to benefit the public at large, while OSI focuses on a limited constituency, the hackers. Hrebenar and Scott (1982, p. 5) describe these two categories as "public interest" and "self-oriented," respectively. Like corporations, nonprofits lobby for open source with an ulterior goal, although for nonprofits the goal is social change rather than material gain. Fortunately for the purposes of this study, the majority of research into lobbying and its effects deals with the lobbying of social change groups.

One of NetAction's stated goals is to "ensure the accessibility and affordability of information technology and the Internet." Its major project since its founding in 1996 has been the "Consumer Choice Campaign," an attempt to educate policymakers and the public

about Microsoft's monopoly abuses and to suggest ways of counteracting the monopoly (NetAction 1996;1 and 1996;2).

In the summer of 1998, in the wake of Netscape's open source release and other news coverage, NetAction board member Judi Clark, a web site designer who is also a board member of Computer Professionals for Social Responsibility and the Conferences on Computers, Freedom, and Privacy, began forming relationships with Raymond and other open source advocates, with the recognition that the movement was in line with NetAction's goal. In addition to being a Microsoft alternative, Clark noticed that open source could contribute to the goal of universal access, the "accessibility and affordability" of NetAction's mission statement, by lowering the cost of a basic computer system.

William Munn defines universal access as the belief that "all people should have affordable access to the national information infrastructure." It has long been a goal of public interest groups, beginning notably in the 1960's when church groups and other concerned organizations campaigned for more access to radio and television broadcasting to increase their usefulness as forms of public expression, and for lower telephone rates for the poor. More recently, both the Clinton Administration and many public interest groups have spoken enthusiastically about the need to insure universal access to the Internet and other emerging forms of high-speed telecommunication, collectively called the National Information Infrastructure. The NII, said some interest groups, "must be inclusive and generous in spirit, ensuring that all segments of our pluralistic society have meaningful access to the telecommunication system" (Munn 1999, pp. 61-72). Although numerous groups are now involved in lobbying on this issue, Clark's outreach to the hacker community made NetAction the first organization to consider open source software as a vehicle for universal access.

Unlike the hacker community, NetAction believes strongly in seeking solutions through government policy. The belief that the policy process is the proper arena for addressing these particular social problems can be considered a defining characteristic of NetAction.

The organization's web site argues that government support and funding are what allowed both the Internet and open source development to flourish, and furthermore that the withdrawal of government support in the early 1990's is what has allowed Microsoft to corner the operating system market and undermine open standards.

NetAction lobbies both policymakers and the public. It distributes two newsletters, "NetAction Notes" and "The Micro$oft Monitor," by email, in addition to writing detailed position papers with the goal of persuading Congress to take notice of their cause. Another tactic, a distinctive feature of NetAction, is their program of "train[ing] activists to use the Internet as a tool for grassroots organizing, outreach, and advocacy" (NetAction 1996;1). The group maintains a Web-based training course for just this purpose, partially in the hope that other activists can support NetAction's cause by learning to use the Internet more effectively.

NetAction's most serious limitations are its size and lack of resources. Funding is extremely limited; only a small amount of income is generated by membership dues. The only significant revenue received by NetAction was a large grant from an anonymous corporate donor, with the requirement that it be used for the anti-monopoly campaign. Repeated rejection for foundation grants has proven to be a major source of frustration for NetAction. Clark believes this is because the foundation world has not yet realized the potential benefits to nonprofits of Internet utilization (Clark, interview). The number of active participants in the organization is also quite small. Director Audrie Krause, board member Judi Clark, and Nathan Newman, who runs the Consumer Choice Campaign, do the majority of all the organization's work. NetAction has an advisory board that, although it is expected to give the organization ideas and direction, has no official responsibilities. The board is potentially one of NetAction's greatest assets, as it contains people from widely diverse fields: a labor organizer, several attorneys, a television producer, a telecommunications policy specialist, and a journalism professor, among others (sixteen in all). However, only four or five of the board members actually provide advice or assistance

on a regular basis. With no employees and an ever-changing volunteer base, NetAction's activities are constrained as much by who is willing to work on each project as by how they are funded.

Fortunately for NetAction, size and funding are not the only determinants of an organization's success. Hrebenar and Scott (1982, p. 31) note that "'Big is powerful' is not an automatic law for lobbying success," implying that the converse is also true. They point to geographic distribution of membership as a potential asset, one that NetAction enjoys. As a "virtual grassroots," NetAction has no central office. Its activities are conducted primarily over the Internet, and its participants live all across the United States, rarely meeting in person. Hrebenar and Scott also mention the wealth, prestige, and education of members as a potential asset, and here too, NetAction is strong (ibid., p. 32). Its board, as described above, contains powerful and well-connected people, potentially increasing NetAction's influence.

Perhaps the most important factor mitigating NetAction's small size is the availability of the Internet as an outreach tool. Storage space for NetAction's website is donated by Clark's firm, ManyMedia. All mailings, including the two regular newsletters, "NetAction Notes" and "the Micro$oft Monitor," are sent by email only. All of the organization's reports and action materials are posted on the Web site as a primary means of distribution. This communication takes place practically for free. If distributing the organization's communications had required printing, photocopying, and postage, NetAction simply could not have existed on its present budget. Browne (1998, p. 91) notes that it is the Internet and other new communication technologies which have made lobbying the public so much more effective.

Finally, NetAction overcomes its size limitations by forming coalitions with other like-minded interest groups. Its relationship to these groups is two-way: generally, one group will take primary responsibility for some policy or educational initiative, and others will provide support and resources with little regard to the distinctions between organizations.

Thus, when a group called the Domain Name Rights Coalition began a campaign for a fairer process of assigning Internet domain names, NetAction helped distribute petitions and organized a letter-writing campaign. In these relationships, NetAction often acts as a "lightning rod," identifying new issues such as open source and publicizing them, so that other organizations can then devote more extensive resources. This was the case with the Consumer Choice Campaign, and to a lesser extent the open source campaign, which were taken up by the Consumer Project on Technology following NetAction's initial identification of the problem. "A coalition with powerful partners may command respect and thus enhance the lobbying usage of the less powerful participants," note Hrebenar and Scott (1982, p. 119). Associating itself with respected lobbyist Ralph Nader has, in this way, helped NetAction get its message through.

To date, NetAction's efforts have resulted in small but not insignificant successes. The Justice Department quoted NetAction's early studies of Microsoft in its opening arguments for the antitrust trial. Thanks to NetAction's having put it, if only peripherally, on the policy agenda, there is talk of using open source as part of the judicial remedy in that trial.

Conclusions

Of the three categories discussed in this chapter, it is clear that nonprofit public interest groups, especially NetAction, will be both the most willing and the most capable groups to seek public policy solutions for the open source issue. These groups see involvement in open source as a means of furthering their goals of universal, equitable access to technology, and to oppose monopoly power. However, because of its resource limitations, NetAction must involve other groups in the process.

Hackers, although generally opposed to government interference, are the prime movers of the open source phenomenon, being those who write software, initiate open source collaboration, and refine techniques. In addition, hackers are increasingly politically aware and willing to lobby for their own ends: persuading businesses to adopt open source

methods. The process of forming open source policy must include the hacker constituency and take their needs into account; if this is done, they can be a valuable resource.

Finally, commercial companies that have begun to oversee or contribute to open source projects see it as a useful business model, rather than as an ideology. Large corporations command prestige and political clout, especially towards a pro-business Congress, such that having these companies as part of a pro-open-source coalition, even if only in name, could increase its effectiveness. However, corporations cannot be counted on for long-term support, as any support they may have for open source is determined only by profit motive and the vagaries of the software market. The characteristics of these three groups are displayed in Table Two.

Table Two: Open Source Advocate Groups

Group	*Goal*	*Assets*	*Liabilities*
Hackers	Corporate adoption of open source, continuation of their culture	Programming skill	Libertarianism
Companies	Profit, serving customers	Financial resources, respect from Congress	Constrained by profit motive
Public Interest Groups	Eliminate monopoly, universal access	Lobbying Skills	Limited resources

Browne argues that the most important determinant of the success of an interest group is not its organizational characteristics, or even its size, but the viability and practicality of the issues it lobbies about. The next chapter will focus on ways of characterizing the open source issue for maximum effectiveness.

Chapter 4
Getting Open Source On The Agenda

> Problem definition is never simply a matter of defining goals and measuring our distance from them. It is rather the strategic representation of situations. (Stone 1997, p. 133)

A necessary first step in any public policy initiative is convincing policymakers and other influential people that government can and should be involved in the issue at hand. Obviously, Congress will not pass a law, nor will the bureaucratic agencies issue regulations, unless the relevant decision makers can be convinced that a problem exists, that a remedy is needed, and that government should implement that remedy. However, the existence of specific problems, and the exact nature of problems, are not universally verifiable truths.

Modern public policy theorists argue that the definition and characterization of problems is itself a political act, and an integral part of every step in the policymaking process. Depending on the way a problem is characterized, it may seem more or less relevant, more or less urgent, to policymakers and the public. This chapter looks at the ways in which open source, and the various public problems it addresses, could be politically characterized for maximum effectiveness.

Political Representation of Problems

An oft-repeated idea among public policy theorists is that public problems cannot be defined and conceptualized by a single rational standard of measurement. Opinions will always differ about whether a problem exists, how severe it is, and whether it is public in nature. Deborah Stone (1997, p. xi) warns against giving public problems "privileged status as universal truths." These authors draw a distinction between abstract *social conditions* and the concrete *public problems* which comprise the day-to-day business of government. Public problems are created by describing social conditions in a way that people identify with, filtering the idea through a body of cultural knowledge and

perceptions. "Onto any social condition ideas, beliefs, values, and interests may be mapped which construct the condition as a problem and, more specifically, a particular kind of problem" (Munn 1999, pp. 16-17).

The language and symbols used to turn a condition into a problem have a direct effect on the way the problem is perceived. An example given by Stone is the issue of welfare spending: When asked about their opinion on government spending for "welfare," 48% of Americans were opposed. However, about the same percent favored "spending on programs for poor children" (Stone 1997, p. 3). Although these two statements describe the same program, the choice of words made the difference in whether it was perceived favorably. It is clear from this example that the way a problem is characterized has everything to do with how people respond to it, both in government and among the public. "Social problems do not come to government fully conceptualized with the labels already attached," says Peters (1996, p. 47). "Policy problems need to have names attached to them if government is to deal with them, and that is in itself a political process."

The way a problem is characterized is a determining factor in which problems are addressed by government and which overlooked. It also affects the number and strength of the activists who get involved in lobbying on an issue. Thus, characterizing a problem is a fundamental tool of activists and policymakers. "Problem definition is strategic because groups, individuals, and government agencies deliberately and consciously fashion portrayals so as to promote their favored course of action" (Stone 1997, p. 133). Different groups may be on the producing and the consuming end of problem characterization at different times. Both public interest groups and individuals within government take part in defining and characterizing issues, and the actions of both are affected by existing characterizations.

Unfortunately for the policy analyst, the strategic use of problem definition makes policy issues all the more difficult to analyze. Since all public problems are filtered through

political characterization and cultural perceptions, an unbiased perspective is impossible. The policy analyst becomes "less a technician and more a politician" (Peters 1996, p. 58).

Problem Characterizations That Work

Various authors have identified those aspects of problem characterization that make an issue more or less likely to result in policy action. These factors can be used by activists and officials to either promote or inhibit a policy action.

Public policy theorists differ in their identification of factors. According to Peters, the perceived severity of a problem is among the most important factors in the creation of policy, and geographic concentration of the people affected by a problem adds to its perceived severity. Those industries which are geographically concentrated, including the software industry, are given all the more attention by government because their problems are more visible (ibid., p. 53). In general, the easier it is to identify the groups or individuals affected by a problem or a potential policy; if "real, identifiable people are the beneficiaries," the issue is more likely to be noticed (Browne 1998, p. 174).

Arguing from a rationalist perspective, Peters writes that problems which cannot be solved through market forces (what economists call *public goods* and *externalities*) ought to be addressed by government, but that only strong proof of market failure will overcome government's prevailing reluctance to interfere in the market. Included in Peters's definition of market failure are activities which the private sector avoids because of high risk (Peters 1996, pp. 56-57). An example of this is the funding of basic research, often considered a high-risk investment by corporations since marketable results are uncertain. A perceived lack of basic research by the private sector could precipitate increased government funding of such activities, such as through the National Science Foundation. The concept of market failure as a policy argument is particularly important to open source, as open source both affects and is affected by monopoly, a type of market failure.

A final criterion addressed by Peters is the availability of a solution. Clearly, a problem is more likely to be addressed by government if the means and technology for solving the problem already exist, making it easier to believe that the problem can actually be solved (ibid., 58). Open source fits this criterion, since it *is* a solution technology for several different problems.

In contrast to Peters's emphasis on the visibility of problems, Browne (1998) concentrates on their compatibility with existing political conditions. According to Browne, the most important factor determining whether a problem is placed on the policy agenda is not subjective popularity or even severity, but whether the problem fits the political climate and existing public policies at the time it is introduced.

> If something fits what's been done before, that issue is easy to integrate into existing institutions. "Modify a policy," say the legal gatekeepers. If something hasn't been done before, and doesn't mess up those public policies that do exist, it's easily added as new policy...Both types make for good issues....If something hasn't been done, and it does threaten to mess with existing directions and funding of public policy, what happens? Generally nothing.
> (Browne 1998, p. 170)

Problems, and policy proposals, are more successful when they are perceived as benefiting a group or issue on which the government is already focusing attention. A perennial example is the business sector: what is good for business is perceived as being good for the economy and national prosperity (ibid., p. 172). If a policy can be characterized as "good for business," it should enjoy support in government. Peters calls this "making a new issue look more like an old issue" (Peters 1996, p. 54). Open source is a business model, and it must be portrayed as a way to increase productivity if it is to benefit from this factor.

If an issue fits the existing political conditions, it is seen by lobbyists as being more likely to pass. Seeking to share in the victory, more lobbyists will be encouraged to join in the advocacy effort. This creates a positive feedback in which politically expedient issues attract more and better lobbyists, which in turn increases the viability of the issue. "The

highly subjective quality of an issue and the otherwise quite observable depth of the lobbying effort reinforce one another" (Browne 1998, p. 168).

On the other hand, issues that aren't compatible with the political climate are doomed to marginality, at best. As with successful issues, Browne believes that what determines a poor issue is not popularity, but degree of compatibility. "These issues aren't necessarily disliked," he writes. "They aren't illogical. They're simply seen as irrelevant at best and politically threatening at worst, but probably both" (ibid., p. 192). Another negative factor for a policy is attempting to interfere in a problem which a certain group already claims to handle best (ibid., p. 207). Excessive interference in business is the classic example of a poor policy by this criterion.

Incompatible policies are not completely eliminated from public discourse, but they are handled on a very limited basis. Key government officials, whom Browne calls *gatekeepers*, act as filters to keep inexpedient issues out of the policy mainstream. These issues may be handled superficially by policymakers, who may perhaps give a rousing message of support or pass a meaningless resolution which claims to promote the cause. This allows them to placate lobbyists and to represent themselves as "truly open to anything" (ibid., p. 194).

Opponents of a particular policy can use the same tools of problem characterization to portray a problem as ill-fitting, disruptive, and inexpedient. They may try to convince policymakers and the public that the problem as stated does not exist, that it is less severe than is commonly believed, or that it is best handled through private means. Table Three sums up the positive and negative factors I've discussed so far.

Table Three: Factors in the Success of a Problem Characterization

	Positive Factors	*Negative Factors*
Rationalist	Visibility of problem	Unpopular
	Concentration	Obscure
	Identifiable victims	Difficult to identify victims
	Market failure	Can be handled in the market
	Solution Availability	No obvious solution
Political	Compatible with political conditions	Doesn't fit existing policies
	Doesn't take something away	Takes away something that a group is used to having
	Beneficiaries already targeted by Congress	Interferes with the free market
	Makes Congressmen look good	

It is important to note that the theories of these authors identify many of the reasons for the success or failure of a policy as being external conditions. Although a policy may be popular, necessary, and have a high potential for effectiveness, it can be stymied by external political factors.

Political Characterization of Open Source: The Economic Argument

Supporters of open source consider it a solution to several different social problems: as a counter-agent to monopoly in the software industry, an aid to education, a vehicle for universal access to technology, and a more sensible economic model for software that will increase companies' productivity. Opponents, mainly Microsoft and its political allies, attempt to portray open source as economically infeasible, of inferior quality, less dependable, and not viable in the long term.

While many of the supporters' claims seem to meet Peters's rationalist criteria for problem justification, including severity, concentration, availability of a solution, et cetera, the same claims may not be compatible with existing politics and policy, and thus will require extensive lobbying efforts to be successful. Associating open source with the monopoly issue may be the most successful tactic in the short term, as this issue is popular

with the public and Congress. The other issues of economic benefit, improved education, and universal access, are not as powerful politically, but nonetheless may provide avenues for future lobbying efforts.

As discussed in Chapter 2, open source is fundamentally about redefining the business model for software, and a major aspect of this change in business model is a new definition of how much intellectual property protection should be retained by software creators. Underlying all of the political claims made by both sides in the debate is both an assumption and a persuasive message about which business model is best, and what distribution of intellectual property rights is proper.

Problem characterization involves the use of metaphors and implicit stories. "On the surface," writes Stone (1997, p. 148), "[metaphors] simply draw a comparison between one thing and another, but in a more subtle way they usually imply a whole narrative story and a prescription for action." Characterizations are designed to appeal to emotion as well as to intellect. They tap basic human ideas of good and evil. Most problem characterizations—the rhetorical material of lobbying—contain an implicit story of some kind. "They are stories with a beginning, a middle, and an end, involving some change or transformation. They have heroes and villains and innocent victims" (ibid., p. 138).

Open source advocates and opponents use stories like these to further their cause. One story archetype used by both hackers, such as Richard Stallman, and public interest groups like NetAction is the story of decline, which goes something like this: "In the beginning, things were pretty good. But they got worse. In fact, right now, they are nearly intolerable. Something must be done" (Stone 1997, p. 138). Stories like this imply a call to action to reverse the decline.

This particular story begins in the computer science research labs of the 1960's and 70's, the period when "things were good." Stallman describes the "software-sharing community that had existed for many years," a group for which source code sharing, like any sort of researcher publishing their results, was taken for granted. This group ultimately

gave rise to all of the companies and basic technologies that make up the computer industry today. The decline of this community, and with it the ideals of openness and collaboration, are attributed by Stallman to the obsolescence of the computers used in these original labs, the growth of proprietary software companies, and the hiring away of many members of the community by these companies in the early 1980's (Stallman 1998, p. 53).

Nathan Newman, writing for NetAction, places the period of decline ten years later, in the early 1990's, when Microsoft rose to industry dominance. His descriptions of "the good old days" include the creation of the Internet through open-source means, and the Unix operating system's rise to prominence. Newman, however, looks to government as both the creator of the original open source community and the architect of its demise. "Open source software, largely funded by government, was the wellspring of...the whole computer industry," writes Newman, and "lies at the heart of how the Internet came into being" (Newman 1999, ch. 1). He points out that most of the original hacker community described by Stallman existed because of government funding, especially through grants from the Department of Defense Advanced Research Projects Agency and through government enforcement of common software standards, such as Unix (ibid., ch. 2). Newman then blames the decline of this community on the waves of deregulation that occurred in the late 1980's and early 90's: "It was the weakening of this government-supervised network of standards in the 1990's that allowed commercial competition over standards to undermine open computing" (ibid., ch. 6). By invoking a story of decline in their publicity materials, Stallman and NetAction are attempting to stir people to action to prevent further decline.

The difference in these two authors' approaches is how they assign responsibility for the decline. Stallman's rhetoric is technologically determinist: the inevitability of progress led to the obsolescence of the university hackers' technology, and ultimately the decline of their community and ideals. Newman, on the other hand, blames government for initiating the decline. The difference is that Stallman assigns responsibility to the impersonal,

inhuman, and immutable forces of technological change, while Newman blames human actors and conscious decisions. The latter argument is more powerful, because it implies that the problem can be corrected by conscious human decision in the form of a policy change. Stallman believes that relief can only come through technological advance combined with the right moral attitudes, and while this has worked for the Free Software Foundation, it is a less compelling argument for non-programmers. In Stallman's worldview, only hackers can solve the crisis, since its cause is technological, and thus he precludes the possibility of cooperation with non-programmer organizations.

The other symbolic story used by open source advocates is one of empowerment and control. This is utilized especially by the Open Source Initiative. The OSI, which was created by a group of hackers to promote open source in the commercial software world, uses a message of improved economic competitiveness in its lobbying efforts. The message they use appears at first to be a simple jump-on-the-bandwagon approach: "Open-source software is an idea whose time has finally come...it's breaking out into the commercial world...Are you ready?" (Open Source Initiative 1998;1). The message implied in the active phrase "breaking out" is that companies should catch on to the movement before it leaves them behind.

This lobbying message could be defined as a story of empowerment and control. Stone describes this type of political message like this: "The situation is bad. We have always believed the situation is out of our control, something we had to accept but could not influence. Now, however, let me show you that we can in fact control things" (Stone 1997, p. 142). In this case, what can be controlled is the business model under which software is developed. The OSI's materials give the message that companies have a choice of business models, that they are not forced to compete on equal terms with powerhouses like Microsoft, that in fact they have the power to change the rules of business to their advantage. This type of story is heartening, because it speaks of increased autonomy and control over one's situation.

Interestingly, Eric Raymond, the author of most of OSI's publicity materials, is quite aware of the necessity for characterizing an issue in a way which the intended targets will agree with. As Raymond thinks in economic rather than policy terms, and because his targets are companies rather than government, he calls this process "marketing" rather than "lobbying," and refers to "positioning" rather than "political characterization." However, the techniques are the same. Characterizing open source in a way which is compatible with business interests was the primary motivation for the creation of the Open Source Initiative. This characterization is the source of contention between OSI and the more idealist Free Software Foundation.

> We have a winning product, but our positioning, in the past, has been awful. The term ``free software'' has a load of fatal baggage; to a businessperson, it's too redolent of fanaticism and flakiness and strident anti-commercialism...In marketing appearance is reality.
> (Open Source Initiative 1998;2)

Businesses, as discussed earlier, cannot let moral or ideological beliefs interfere with the bottom line. A lobbying message directed at companies should avoid a strong moral stance and stress economic value. The OSI's message does just that: "We think the economic self-interest arguments for Open Source are strong enough that nobody needs to go on any moral crusades about it" (ibid.).

Basing their argument solely on economic and competitive grounds has been a reasonably successful strategy for the OSI, with several prominent companies working open source methods into their businesses. Although part of the hackers' motivation for the OSI campaign is ideological, based on the belief that the open source business model is more equitable, this fact is not explicit in their lobbying materials.

These two symbolic stories, decline and control, form two facets of the current open source lobbying effort. As Stone points out, they are highly interrelated.

> Stories of control offer hope, just as stories of decline foster anxiety and despair. The two stories are often woven together, with the story of decline serving as the stage setting and the impetus for the story of control. The story of decline is meant to warn us of suffering and motivate us to seize control.
> (Stone 1997, p. 144)

Although it has proven successful, on a limited basis, in the corporate world, the message that businesses can be made more competitive using open source will not be a persuasive issue when directed towards Congress, as it attacks two cornerstones of American thought: that companies should make their own decisions about how to operate, and that wealth is generated through well-defined ownership of the goods one creates. The proprietary software model has historical precedent, as it arises from ideas of intellectual property formed in the publishing and music industries. To corporate managers and Congress, the proprietary model is the software industry's reason for existence and its economic foundation. Asking policymakers' help in changing that model would seem to them like a plan to corrupt the industry and ruin its profitability, and this will hardly be popular. In addition, if Congress were to design policy based solely on the "superior business model" characterization, they would seem to be telling the software industry how to conduct its day-to-day business, and this too is a political mistake.

Associating Open Source with Antitrust

The one circumstance under which government will occasionally leave behind its laissez-faire position towards business is strong evidence of market failure, such as a monopoly. Although the free market is a beloved tenet of American political and economic thought, the antitrust laws, which are often perceived as government's heroic struggle against heartless monopolists, hold a position almost as hallowed. "Antitrust law has become an icon in our society," says Philip Areeda (1992, p. 32).

For a number of reasons, associating open source with antitrust policy may be the most effective characterization of the issue in the short term. First of all, the issue is very visible right now, with extensive coverage of the Department of Justice's antitrust lawsuit against Microsoft. With help from the media, the question of anticompetitive practices on the part of the giant software company has gained a firm place in the public consciousness over the past several years. Newspapers regularly proclaim how "Microsoft has been clearly shown

to exercise monopoly power in operating-systems software for personal computers in ways that coerce competitors and clients, depriving consumers of choices" (New York Times 1999). In a representative two-week period, January 1st through January 15th of 1999, major U.S. and European newspapers mentioned Microsoft in 515 articles, and of these, 72, or 14% of the articles, called Microsoft a monopoly[13].

As a result of this media attention, Microsoft has become strongly associated with monopoly. It has become the company we love to hate. Websites like "the Anti-Microsoft Association" and "The Microsoft Boycott Campaign"[14] proliferate on the Internet. One indexing site, "The International Anti-Microsoft Network[15]," lists 256 such sites. NetAction director Audrie Krause calls Microsoft CEO Bill Gates "the modern-day equivalent of a Robber Baron (NetAction 1996;2)," attempting to create associations with turn-of-the century monopolists such as Andrew Carnegie who prompted the creation of antitrust laws. Congress itself has not been immune to this popular sentiment. Senate Judiciary Committee Chairman Orrin Hatch, Republican of Utah, has vociferously accused Microsoft of "excluding competitors [and] stifling innovation" (Wilke 1998). The Department of Justice and the federal court system have obviously placed the software antitrust issue on their policy agendas, committing resources to a high-profile lawsuit against Microsoft. Monopoly is clearly an important issue for the public, the media, Congress, and the judiciary.

Associating open source with the popular issue of antitrust should help put it on the policy agenda. To a degree, this association has already taken place. The way the media, and presumably many interested citizens, define open source is specifically as an opposition to Microsoft and its monopoly. The San Francisco Chronicle, for example, predicted that "the biggest challenge facing Microsoft may not be coming from the Justice

---

[13] These statistics were obtained from a search on the Lexis-Nexis Company News database, comparing the results of a search for "microsoft" with one for "microsoft and monopoly."

[14] http://users.aol.com/machcu/amsa.html
http://msbc.simplenet.com/

[15] http://www.webring.org/cgi-bin/webring?ring=antims&list

Department or Sun Microsystems. Rather, it's in the form of a free operating system called Linux" (Einstein 1998). Out of 73 articles in major newspapers in 1998 and 1999 that dealt with open source, 31 of them mentioned Microsoft, and 11 referred to monopoly or antitrust[16]. Continued efforts by NetAction and other open source advocates to portray open source as a tool of antitrust policy should increase the likelihood of policy that supports open source.

Tying open source to the monopoly issue could benefit policymakers, since open source could serve as an effective antitrust policy. This association fits Browne's criterion that "good issues can be seen as good for government" (Browne 1998, p. 172). Policymakers support issues that make them look successful.

An Aid to Universal Access

A third area of public policy with which open source could be associated is the "universal access to telecommunications" agenda, although to date this association has not been made explicitly. Although perhaps not as powerful an issue as antitrust, ensuring access to communication technology by the poor, remote, and underprivileged is a stated goal of the Clinton Administration's telecommunications policy and a major goal of many public interest groups. As part of the lobbying message, showing how open source could be used to further the goal of universal access is another political characterization which could be helpful to the cause.

The ideological basis of universal access is that communication and access to information are fundamental to human progress and self-improvement, and that a lack of access to modern communication technology would limit political participation and economic potential.

> Access represented the key to unlocking the benefits and promises of the information infrastructure....A lack of access for the individual...would

---

[16] Lexis-Nexis search, see note 13. The search criteria compared were "open source," "open source and microsoft," and "open source and (monopoly or antitrust)"

> create a serious handicap to full participation and maneuverability in daily economic and social life. On a societal level gaps in access were understood as leading to a digitally divided nation. One with a weakened fabric and democratic foundation.
> (Munn 1999, p. 33)

In the context of the recent 'information revolution,' it is believed that much of the give and take of information which constitutes political discourse, as well as economic information such as job listings, will move to the Internet and other high-tech forms of communication. If the basic means of self-expression and political participation are too expensive for a large percentage of Americans, their well-being will suffer. Discussion of "information haves and have-nots" is common among universal access interests.

Presumably, universal access campaigns have existed since the invention of the printing press. In recent years, interests have shifted from citizen access to broadcasting in the 1960's, to cable TV rate regulation in the 1970's, to media accuracy watchdogs and anti-television-violence groups in the 1980's. Over the past ten years, telecommunications has become a more important issue for public interest groups. The immediate motivation for this interest was the break-up of the AT&T monopoly and waves of deregulation, which activists feared would lead to telephone rate increases (ibid., p. 66-68). Most recently, the growth and increasing importance of the Internet has caused numerous groups to take up the access issue. William Munn's dissertation on universal access focused on twenty such groups, among which are the Alliance for Community Media, the American Association of Retired Persons, the American Library Association, the Electronic Frontier Foundation, and the United States Catholic Conference (ibid., p. 9).

The actions of these groups were undoubtedly part of the impetus for the federal programs Lifeline Assistance and Link-Up America, that lower the cost of telephone service for the poor (ibid., p. 69). They also played a role in shaping President Clinton's telecommunications policy in the early 1990's. A statement of policy released in 1993, "The National Information Infrastructure: An Agenda For Action," mentioned "universal service" as one of the Administration's goals:

> A major objective in developing the NII will be to extend the Universal Service concept to the information needs of the American people in the 21st Century. As a matter of fundamental fairness, this nation cannot accept a division of our people among telecommunications or information "haves" and "have- nots." The Administration is committed to developing a broad, modern concept of Universal Service -- one that would emphasize giving all Americans who desire it easy, affordable access to advanced communications and information services, regardless of income, disability, or location.
>
> (U.S. Department of Commerce 1993, ch. 1)

Using open source as part of a universal access lobbying effort is an obvious choice, because it lowers the cost of a computer system. Universal access is typically characterized by interest groups as a problem of physical access to communication networks, asking the question, "do people have a phone and do the wires reach their house?" Furthermore, a study conducted in the mid-1980's showed that the primary reason why people cannot afford telephone service is the startup costs of equipment and installation, not the monthly service costs (Munn 1999, p. 69). Presuming that the same condition is true for access to the Internet, open source, which would allow for the purchase of a functioning computer system with almost no software costs, could have a dramatic impact on the number of lower-income households who could be connected to the Internet from their home, school, or library.

An advantage of associating open source with universal access is the broad interest group support that the latter issue receives. Writes Munn, "one of the more interesting aspects of the access debate was the extent to which organizations with interests typically not related to communications policy enter into the policy arena to press their claims (ibid., p. 77). If open source can be portrayed as a powerful way of promoting universal access, diverse and widespread public interest groups may be convinced to lend their support to the open source movement.

Although universal access appears prominently on statements of Administration policy, little concrete government action has occurred. One policy, called the *e-rate*, provides discounts on Internet access of up to 90% to libraries and schools. Hailed as a positive and needed initiative, the e-rate attracted 30,000 applicant schools and libraries in 1997, and

32,000 in 1998, some $2 billion in aid based on the program's rules of entitlement. However, only $1.2 billion was ultimately allocated, meaning that many institutions received no funding (EdLiNC, 1999).

Aside from halfhearted universal access policies like the e-rate, the results of government policy have been mainly to turn over control of information infrastructure, and its social responsibilities, to private industries. Thus, universal access may not be the most politically opportune issue with which to associate open source at the current time. However, universal access is tenacious: it has been part of government policy since the Communications Act of 1934 and has withstood periods of inactivity in the past. It is likely to reappear as a viable issue sometime in the future, as more information about the social and political effects of the information revolution comes to light. If open source advocates have combined forces with universal access groups by that time, both will benefit.

The different political characterizations of open source are summarized in Table Four.

Table Four: Political Characterizations

Idea	*Pros*	*Cons*
Open Source helps businesses succeed.	Congress is pro-business	Interfering with fundamental business model
Open source eliminates monopoly	Lots of public attention	Hasn't been proven
Open Source promotes universal access	Many potential interest group allies	Low Congressional attention to issue

Conclusions

This chapter has looked at several tactics that are or could be used to characterize open source as a political and economic issue. The Open Source Initiative's campaign to "sell" open source to business based on its economic merits and NetAction's linking of open source with the monopoly debate attack the problem from both ends by increasing the use of open source and potentially decreasing the ability of certain companies to oppose it.

These characterizations, though they come from two very different political worldviews, are not incompatible. On the contrary, the combining of economic and social arguments for open source strengthens the overall lobbying message. These two arguments are more persuasive when they are integrated, and this integration can only be accomplished through coalition-building between hackers and public interest groups.

Chapter 5
Policy Tools: What the Government Can Do

> Effective maintenance of a community or pursuit of common goals cannot possibly be accomplished by governing every action or decision of individuals and organizations. Societies rely instead on broad structures and rules that will have a "multiplier effect," shaping people's behavior without continuous and specific directions.
> (Stone 1997, p. 260)

Having discussed methods for bringing open source to the government's attention, I now turn to my final point of inquiry, the question of what the government's role should be. Which of the numerous powers of government should be applied to the task of increasing the production and use of open source software, while satisfying all of the groups involved and not causing the movement to bog down in political wrangling? The answer lies in choosing policy instruments which are compatible with existing political conditions, which benefit policymakers as well as the affected groups, and which don't take anything away from any politically significant group that they're used to having.

The most basic—and most naïve—policy tool is direct regulation of the software industry. For example, mandating that all software companies, or some companies, or all makers of a particular type of software, release their product under an open source license. If open source is a good move, if it has economic and social benefits, then why shouldn't everyone be using it? Perhaps all operating systems should be open source, so that anyone can write completely compatible application programs? The first obvious problem with this approach is that Congress would never enact such a policy. Although the government often tells companies to change their ways, for example by mandating fair treatment of employees or outlawing unfair pricing policies, it is only under the most extreme of conditions that government will force companies to change their fundamental business model, the terms of the transaction from which they generate revenue—and extreme conditions are extremely hard to prove. Requiring companies to shift their revenue source from copyrighted software to software-related services is too radical an intervention in the industry to be seriously considered by policymakers.

The second problem is that even if Congress did pass such a decree, it could destroy open source rather than promoting it. The hacker community that creates most open source software today is primarily libertarian and would not look kindly on direct government regulation. According to *Linux Journal* editor Phil Hughes, "they participate because no one's telling them to" (Hughes, interview). If laws or regulations mandated where and how open source software was to be created, as would surely happen in any attempt to force open source licensing on the industry, the majority of current open source programmers would simply pack up and go home, leaving the movement without its productive core. "Don't mandate [open source]," said Hughes, "just make it economical and then watch what happens."

Hughes's point is an important one: companies are motivated by profit. "If they're not opening their code, it's because they have economic reasons not to." A successful policy, rather than mandating or directing the use of open source, would make open source fit companies' economic interests, and make them *want* to switch over. There are numerous possible ways of doing this, including government purchasing, research and development spending, education programs, financial or tax incentives, and competitive regulation.

This chapter will focus on two of these policy areas in depth: government purchasing policy, or procurement, and federal involvement in information technology through research grants, standards maintenance, and education. The idea behind procurement policy is that the federal government, being a very large, wealthy organization with large equipment needs, can have a powerful effect on the price and availability of products, and on the behavior of producers, by strategic use of purchasing (in this case, information technology purchases). Research and development funding shapes an industry's long-term course by selecting the areas in which basic research is conducted, and by supporting certain groups, social structures, and ideologies within the research community. Standards, the common languages of computer hardware and software, are vital to the health of the industry.

The rationale for these areas of focus was to identify particularly illustrative examples of policy tools that (a) are politically feasible, (b) complement past and present lobbying efforts, and (c) provide useful illustrations of possible implementation pitfalls. That said, the choice of policy tools was also influenced by the availability of data and the interests of the researcher. The conclusion of this chapter will show how the principles of policy implementation discussed with regards to procurement and R&D policy can be generalized to other policy tools.

The other policy options that I will look at in less depth are financial incentives, such as tax breaks for open source businesses, and protection from competition, including possible judicial orders handed down in the Microsoft case.

The analytical method used in this chapter is to compare potential open source-related policies to similar policies enacted in the past. Since open source as a political phenomenon is new and untested, no direct data on the effectiveness of potential policy tools is available. By examining the successes and failures of similar policies, and then identifying their similarities and differences with regards to open source, I will draw some conclusions about the potential effectiveness of these policy tools.

Procurement Policy and the case of solar cells

Procurement is the use of government purchasing contracts to affect the market for a good. It is potentially a very powerful policy: the federal government is the largest consumer of computers and software in the world, so its purchasing choices are bound to have a great effect on the price and availability of products. The government as a whole has apparently neglected the policy implications of its software purchasing, and the result is a certain aggregate hypocrisy: while the Justice Department and the courts pour resources into a lawsuit of Microsoft on antitrust grounds, the executive branch agencies primarily run their computers using Microsoft's Windows NT operating system, giving considerable

financial support to the same monopoly that other agencies are committed to restraining. A procurement policy for open source would rectify this contradiction.

The related case which I will use to illustrate the potential and pitfalls of procurement policy is a 1976 plan by the Federal Energy Administration to promote photoelectric solar cells as a means of renewable energy production[17]. This program came in the context of the national energy crisis of the mid-1970's. President Jimmy Carter's first major presidential initiative was to propose a plan for reduced dependence on foreign oil. However, this plan focused mainly on energy conservation, and did very little to attack the underlying problem: the depletion of nonrenewable energy sources like oil. This omission led some Congressmen to begin looking into alternative energy sources, resulting in the passage of the Energy Conservation and Production Act in August of 1976, which mandated the development of strategies for "widespread commercialization of solar energy" and created a "Task Force on Solar Energy Commercialization."

The innovative idea of this task force was to depart from the standard method of technological development, in which a technology is researched, prototyped, tested, and demonstrated, and finally given over to the market for production. In this model, economic aspects of the technology are only considered once it has been built. The FEA task force took the opposite approach: concentrating first on the *market* for solar power. The technology was already available: photovoltaic cells, thin silicon wafers that produce electricity when struck by sunlight, had already been developed for the space program. However, they were far too expensive to compete with the coal and nuclear power that gives light to people's homes. According to the old way of thinking, what was needed was a technical breakthrough that would reduce the cost of solar cells. What the task force did instead was set out to reduce the price through the use of economies of scale rather than a technological leap.

---

[17] The description of the solar cell commercialization plan, except where otherwise noted, is taken from (Commoner 1979, pp. 33-38)

Economies of scale presented a "chicken and egg" problem for technologies like solar cells: only large demand would give manufacturers the incentive to expand their plants and production facilities to a more efficient point, thereby reducing the unit cost of solar cells, yet this demand was not forthcoming because of the existing high prices. What was needed, decided the task force, was for government to make a series of investments in solar cells, allowing production to expand and economies of scale to take effect.

This "administrative ploy" had been used successfully by the Defense Department between 1965 and 1975 to lower the cost of integrated circuits, the "computer chips" that are found today in almost every electronic device. In this ten year period, integrated circuits fell in price by a factor of 100 and their performance increased by the same amount (Federal Energy Administration 1974, p. VII-22).

The key to the plan for solar cells was to phase the technology in, first installing the cells in areas where power was expensive and solar cells were already economical, then, as the price fell, introducing them into more and more markets. The catalyst for this process would be a government purchase, but once started, it would happen through the market, without intervention. Solar cells would be installed to replace gasoline generators at military installations, then gas generators in rural towns, then street lighting, then airport emergency lighting, and finally, after about five years, the price would be low enough to compete for the lighting of homes.

The economic rationale is compelling: if government makes the initial investment, it will trigger a series of industry expansions and price reductions, until the technology is ubiquitous. Unfortunately, the solar cell plan did not work as expected. The Carter Administration dragged its heels, opposing legislation for a purchase plan. The plan that was eventually passed into law allocated $98 million for solar cell purchases rather than the $400 million requested by the task force. This drastic reduction would have doubled the time in which the price of solar cells could be expected to fall to competitive levels. Even this reduced plan was never carried out, however, because President Ronald Reagan took a

vocal and unapologetic stand against all renewable energy funding, giving his advisors a "license to kill" nearly all solar energy programs in favor of coal and nuclear power (Hempel 1982, p. 209).

The story of the solar cell commercialization plan and its demise contains lessons for open source and a possible procurement policy. In many ways, the circumstances are similar. For one thing, both policies are responses to a crisis situation. While the "software crisis" is neither as severe nor as visible as the energy crisis of the 1970's, it is nonetheless a pressing problem in both government and the private sector. The Year 2000 problem has brought a sense of increased urgency to this situation. Another similarity, and a necessary precondition to this procurement policy, is that the technology is already available and avenues for further improvement are visible. Open source provides a pre-existing solution to widespread reliability problems, and it has the potential for even more reliability, security, and interoperability if a policy such as procurement raises interest in the method. As with solar cells in the mid-1970's, "there do not appear to be any critical technical problems associated with the application...(Federal Energy Administration 1974, p. VII-16)"

The social aspects of solar energy and open source are similar as well. Both are decentralizing technologies, under the control of users rather than producers. Along with this decentralized nature comes more accountability to individuals and communities, a focus on individual needs rather than mass solutions, and an emphasis on labor rather than capital (Worthington 1984). Like solar energy, open source can be produced closer to its intended recipients, at least in the logical sense of focusing on the needs of a particular individual or group.

The most notable difference between open source and solar cells is that open source is a business model, not a specific product or technology. The difference is that for open source, economies of scale have more to do with gaining consumer confidence than with lowering price. Open source is already more affordable (in terms of total cost of

ownership) that proprietary software in most areas, but what it lacks is a wider pool of users and producers and the consumer confidence that comes with widespread use. The designers of the solar cell plan acknowledged this issue:

> Although the conditions may become propitious for such business ventures, the uncertainty of market projections may delay these venture-decisions. This would cause a serious deterrent to the participation of many competent...manufacturers. hence, an important decision should be made early in the program to encourage companies to enter the field.
> (Federal Energy Administration 1974, p. VII-48)

For open source this is not only part of the issue, it is the entire issue. The phases of a government procurement plan for open source, rather than beginning in areas of high cost, would begin in areas of high consumer confidence: Web servers, file servers, and other "back office" applications where open source is already widely used and trusted. As more open source software is installed for these uses, the companies that make them will grow, and other companies will be encouraged to go open-source as well, applying their brand names and support services to improving the public perception of open source. Eventually, consumer confidence would rise enough for open source to break into more desktop computing applications in business, and ultimately in homes.

The catalyst for this progression could be a policy like this: "All operating systems purchased by Federal agencies must conform to the OSI's Open Source Definition." There is historical precedent for such a policy. In 1986, seeking to cut costs by reducing the inefficiency of heterogeneous networks and incompatible computer systems, the government required any computer company that bid on federal contracts to install the Unix operating system on their machines (Hall and Barry 1990, p. 105).

Unlike solar cells, for which the government itself had only a limited need, every branch and division of government uses information technology, making them all potential buyers of open source. This should make a procurement policy more amenable to Congress. While buying solar cells at high cost for a small number of military facilities may have seemed wasteful to policymakers, a change in purchasing policy for something that all sectors of government are buying anyway—computers and software—should be much

easier to justify. Furthermore, while the solar cell plan involved government purchases of the technology while the cost was still high, open source will provide from the outset a substantial cost savings for government. In the absence of restrictions on copying, a government agency could buy a single copy of some necessary software and install it on every computer in the office. It is clear that this policy would benefit the government directly, as well as achieving the desired effect on the economy, and such "win-win" policies are popular with Congress.

Ripley and Franklin divide all policies into four types. Open source procurement would be classified as a *distributive* policy, the easiest type to implement (Ripley and Franklin 1986, p. 73). This type of policy generally sees a low level of conflict and little direct attention from Congress unless a complaint is raised (ibid., p. 84). The only groups from which this policy takes something away are proprietary software companies that are unwilling to switch to open source. These companies will undoubtedly oppose a policy like this. Microsoft is particularly likely to protest such a policy, though being on trial under antitrust law should serve to delegitimize their complaints. All other groups, government, open source producers, and ultimately the public, have something to gain from this policy, so it is reasonable to believe they would support it. The hacker community sees procurement as the sort of government policy it is comfortable with; Eric Raymond in particular endorsed this course of action (Raymond, interview).

A potential problem with the procurement policy is the political strife that would undoubtedly come over how open source is to be defined. Different groups define open source very differently. While the OSI's Open Source Definition seems like a good general framework that covers a variety of software licenses, wrangling over its individual points will be inevitable if they are given the support of government policy. A way to avoid this would be to set the terms of government purchasing by some more objective standard, such as reliability. All government offices want their software to be reliable and fault-tolerant, so a possible policy would require software to pass certain quality control tests, or perhaps, to

crash less than a certain number of times per week under normal usage. If open source does in fact produce more reliable software, this policy should favor open source. Other criteria are possible as well in which open source has an advantage, including security, compatibility with other products, conformance to Internet and other standards, and the most obvious criterion, lowest cost, which is almost always a popular policy choice.

The advantage of this approach is that it doesn't explicitly favor open source, so it should meet with less resistance from proprietary software companies. It does not imply any sort of ideological commitment to open source on the part of Congress or the executive branch, making it less politically charged. The political position it implies is pro-competition rather than competition-restraining. The downside is that it is entirely possible for proprietary software companies to fit these criteria as well, if their software is up to the task, in which case the goal of increasing open source production would not be achieved.

The biggest lesson to be learned from the solar cell example is that, while the economics of procurement policy are powerful, once such a policy is proposed it must be followed through completely to achieve the desired results. Half-measures and inadequate funding won't push open source over the threshold of widespread acceptance. The initial push created by policy must occur quickly, and both producers and potential consumers outside of government must be encouraged to get involved early on, in order for the economic effects begun by the policy to become self-sustaining quickly. This will insulate the plan from changes in political conditions, such as what happened when Reagan replaced Carter. If these issues can be worked out, procurement could prove to be the most effective type of policy for open source, as the economics are compelling and all involved parties stand to gain.

Lessons from the Internet: R&D Funding and Standards [18]

Both the Internet and open source software have their origins in the government-funded laboratories of universities and defense contractors. The history of government patronage of the Internet sheds light on the ways in which government research spending affects the course of industry, and suggests several policy tools that could be created or modified to promote open source both as a business model and as a philosophy. The government, especially the Defense Department's Advanced Research Projects Agency and the National Science Foundation, used a combination of grants, organizational structures, education, collaboration with industry, and standards maintenance to shape the Internet into a powerful tool for both commerce and democracy. These agencies, for the most part, struck an optimal balance between government oversight and educational and industry autonomy, a model that could benefit open source as well.

The Internet began life in 1970 as ARPANET, the Advanced Research Projects Agency's network which linked universities and defense contractors across the country. Its initial purpose was to aid collaboration among the top computer science researchers of the day, allowing them to share results and avoid duplicated work. The character of ARPA was shaped by its founder, President Eisenhower, who distrusted the military but liked the scientific community. "He found scientists inspiring—their ideas, their culture, their values, and their value to the country" (Hafner and Lyon 1996, p. 15). Eisenhower wanted the government's research dollars put into a community that valued flexibility, inventiveness, and sharing over regimentation, centralized authority, and cozy relationships with business. He hired ARPA directors who shared this view, creating an organizational culture that was "freewheeling, open to high risk, agile," and the agency's "relatively small size allowed the personality of its director to permeate the organization" (ibid., p. 22). The values articulated by Eisenhower and the early ARPA directors led directly to the

---

[18] The history of the Internet in this section is derived from (Kahn 1994), and (Hafner and Lyon 1996)

democratic nature of the Internet. Research and development funding is not impartial—it can be used to favor those researchers who hold certain values.

The Clinton Administration has recently renewed its commitment to funding information technology research. As part of the "Information Technology 2" initiative, the administration announced a 6% increase in funding for this research through the National Science Foundation, with 60% of this funding to be directed to universities. Along with the actual funding, a policy that would help open source would be to promote the values of code sharing and open development processes among research grantees. As these values are already common to universities, targeting them for funding is a powerful move. Funding universities while encouraging open source methods has the added bonus of empowering the next generation of open source programmers—computer science students. Access to source code and participation in open software development projects is the best hands-on computer science education there is, so with no extra investment, this type of policy improves education and extends the pool of potential open source developers.

The other major role that the government played in the evolution of the Internet was in the maintenance of standards. The most successful strategy, and the one that led directly to the current set of universal Internet protocols has been to encourage the development of standards by industry consensus, and help to maintain those standards arrived at by industry.

The first central coordinating body for what was then the ARPANET was called the Internet Configuration Control Board. Although it was a government-created entity, the board was staffed with members of the research community that ARPA supported. The government also contracted groups to assign names and numerical addresses to computers on the network and maintain a database of those addresses. Over time, these organizations lost their government affiliation. The ICCB became the Internet Activities Board, which changed to the Internet Architecture Board in 1992 and became part of the nonprofit Internet Society (Kahn 1994). Throughout this time period, and despite its plans to give up

control of the network, the government instilled in these organizations the philosophy that open standards, created by consensus in a public, inclusive process, were the key to industry success.

Open source and open standards go hand in hand[19]. Open source developers benefit from the availability of published standards to which other software and hardware makers adhere, and once a standard is written into an open source program, the standard is plainly revealed in that program's source code. Renewed interest by government in maintaining open computing and communications standards favors the open source model.

The difficulty in this type of policy is setting the best level of government control. On one hand, standards set directly by the government don't work well in the computer industry. The best example of this is the International Standards Organization, a U.S. government-controlled body, and its Open Systems Interconnect protocol series. Created on paper without a working software implementation, the OSI[20] protocols were declared official government standards in both the U.S. and Europe. However, they failed to compete with the TCP/IP protocols, still in use today on the Internet, which had been developed by the Internet Activities Board with government funding, rather than government fiat.

The other extreme is also undesirable: proprietary protocols such as those favored by Microsoft. If a single company controls the rules by which computers interact, that company has the power to exclude others from the market, possibly preventing innovation by other firms. A complete absence of government enforcement of standards allows this situation to occur. It is for this reason that Robert Kahn argues for a continued government interest in the Internet governing bodies:

> A further concern is the viability of any entity that has no individual or organization with overall responsibility for its evolution...history has shown that a government role was necessary to make it happen in the first place. What guarantees are there that the same degree of vitality in its future

---

[19] See Figure 2, in the Appendix

[20] Not to be confused with the Open Source Initiative

> evolution will take place if market forces alone determine what new capabilities are added to the Internet?
> (Kahn 1994)

This concern applies to open source as well. As with procurement policy, the challenge in standards enforcement is to reward openness and conscientious compliance without favoring some standards over others or overly politicizing the process. The optimal policy may be to maintain a government interest, rather than government control, in standards-setting bodies such as the IAB. Encouraging companies to label their products with the standards with which they comply, and threatening sanctions against those who advertise compliance falsely, would be an effective and relatively neutral way for the government to use its influence to support open standards.

This type of policy would be characterized as *protective regulatory policy* by Ripley and Franklin. This type of policy "is designed to protect the public by setting the conditions under which various private actives can occur" (Ripley and Franklin 1986, p. 76). Protective regulatory policy is usually more difficult to implement than distributive policies such as the procurement policy mentioned in the last section, but easier than redistributive policies that take away from some groups and give to others. The difficulty in protective regulatory policy is that its beneficiaries—the public—are vague and hard to define, while the groups it restricts—large companies—are powerful (ibid., p. 88). Although not as politically facile as procurement policy, standards oversight is compatible with the government's earlier policies towards Internet standards, and the wildly successful result of those policies provides evidence of their effectiveness.

Other Policy Tools

A third area of policy solutions which I will mention only briefly is financial incentives such as tax relief. In this case, the government could offer tax credits for companies developing open source. Considering the network effects theory, a little financial help of this sort could give key open source products a "first-mover advantage" in the market,

leading to long-term dominance. Unfortunately, this is subject to many of the same political pitfalls mentioned above, concerning how open source is to be defined for the purpose of giving out the tax credits. Any such definition would be seen by open source producers as excessive government interference in the open source process, which will discourage many programmers. Hackers, with their belief that open source is economically superior, would likely be offended by financial incentives, as they carry an implied message that open source is weak and needs artificial help to survive in the market. "I don't think Linux and open-source should get preferential treatment, just fair treatment," said Phil Hughes (Hughes, interview).

A final policy tool which is extremely relevant right now is the sanctions to be handed down in the Microsoft antitrust trial. Whether agreed upon in a settlement or handed down unilaterally by the judge, some sort of penalties are likely to be imposed against Microsoft for its anticompetitive behavior, and that penalty could serve as a pro-open-source policy if it is carefully considered. Potential penalties may include breaking up Microsoft into smaller companies, either vertically, creating mini-Microsofts with competing product lines, or horizontally, by separating the operating system and application divisions into separate companies. At the very least, the penalty will address the concerns that prompted the lawsuit: forcing Microsoft to desist from unfair terms in its licensing of software to computer manufacturers, preventing predatory pricing of Web browsers and other software, and preventing the "bundling" of Web browser software with the operating system. Any curtailment of Microsoft's monopoly tactics creates more market potential for all of Microsoft's competitors, including open source competitors. However, the judicial sanction could be more proactive than this. Restraining Microsoft would create a void in the industry but would not dictate the best way to fill it. NetAction's Newman points out that

> many critics of the Microsoft suit raise reasonable concerns that a purely negative, restrictive approach to punishing Microsoft might inhibit innovation at the company without necessarily creating a viable competitor. Promoting open source software is the positive policy option that the government should employ.
> (Newman 1999, ch. 10)

Proactive policies set down by the judge could include any of the policy options discussed in this chapter, especially the increased enforcement of industry standards. Richard Stallman, of the Free Software Foundation, has allegedly suggested that Microsoft be forced to release the Windows source code. However, this policy is doomed to failure for the reasons outlined at the beginning of this chapter—mandated open source will be useless at achieving beneficial social changes. Judge-made policy has the additional bonus of being perceived as more legitimate and apolitical than congressional policy. This is because the public perceives judges as being more trustworthy than legislators (Areeda 1992, p. 33). In any case, the window of opportunity for open source activists to affect the trial outcome is quite small, so this form of policy requires immediate action. Table Five shows the policy tools I've discussed in this chapter.

Table Five: Policy Tools

Policy	*Pros*	*Cons*
Mandated open source	Cuts to the heart of the issue	Wouldn't pass Congress Wouldn't produce results Direct interference in the economy
Procurement	Inexpensive sound economic reasoning pro-competition something government needs anyway	Defining open source may create conflict
R&D Funding	Allows giving support to a group or ideology Improves education as well as the economy Priority for Clinton Administration	Could be expensive
Standards Enforcement	Successful in the case of the Internet Preserves industry autonomy	May create conflict over who sets standards
Judicial Remedy	Carries judicial legitimacy Fast	May interfere in free market too much Choice of policy up to the judge

Conclusions

The recurring theme of this chapter is that any policy solution chosen must be one that all parties concerned can live with, because any one of the involved parties could block the policy's implementation at some stage. A policy that says "this is how to do open source, this is how it's defined," will alienate programmers, stripping the movement of its productive core. On the other hand, purely economic policies that divorce open source from its social values of consumer empowerment and universal access would leave public interest groups—the movement's political lobbying experts—with no motivation to participate. Even proprietary software companies, although the antitrust trial may put a limit on the effectiveness of their complaints, must be kept satisfied, mainly by the government's agreeing not to meddle with their fundamental business model.

Another lesson to be learned from exploring policy options is that since relatively "easy" policies are available, those which give tangible benefits to all parties concerned, don't take away what groups are used to having, and appeal to Congress, there is no reason to waste resources pursuing policies which, no matter how beneficial, are not politically expedient.

Chapter 6
Where is Open Source Headed? Conclusions and Recommendations

> What does "programmer" mean? Do we mean only those who build modules in languages like C, C++, or Pascal? Or does a Unix shell programmer qualify? How about a Macintosh user who chooses and assembles off-the-shelf objects like word processors and spreadsheets to build a custom solution to personal computing needs?
> (Cox 1996, p. 79)

Whether or not open source is a commercial success, its message is both compelling and a little frightening. The message of open source is this: "Software isn't magic, and we'll show you its inner workings to prove it. You can write software yourself, or hire a friend to write some for you. Your choices are in fact unlimited and should be dictated by your precise needs." Open source allows anyone to be a "programmer," and redefines the term to make this statement true.

Open source allows the understanding that all of these levels of abstraction are fundamentally the same. It blurs the distinction between programmer and user. Open source brings to light the fundamental notion that computers are tools, and that ultimately what users need and want is not software, but tools to accomplish the work they need done in the most efficient way. This is a subtle distinction, but an important one. Proprietary software, since it is written to be used by many people in more or less the same way, is *software*, which may fit a particular user's needs, tastes, and style to varying degrees; it is one-size-fits-all. Open source, once the infrastructure is in place to allow users to obtain a system of carefully chosen, linked software components built from standardized parts but modified and assembled specifically for that person, becomes a *tool*, allowing the user to forget more completely about the computer and concentrate on the task. Open source challenges our assumptions about computers: that the only way for software to be produced is for the producers to retain all rights in their products, and that "the only important thing about software is what jobs it allows you to do" (Stallman 1999, p. 54).

The other assumption that open source challenges is that technology is inherently dehumanizing, a centralizing force, amenable to control of large, powerful organizations.

On the contrary, technology can empower the individual, provide choice and control, and be no less advanced or relevant. This is similar to the social message of decentralized renewable energy, economically productive conservation and recycling, and other "green" technologies. Empowering technologies could have tremendous political impact. "Stories of control are always gripping because they speak to the fundamental problem of liberty—to what extent do we control our own life conditions and destinies?" (Stone 1997, p. 142).

Challenging our assumptions is a frightening proposition. The business world is built on these assumptions, and business is what generates wealth and economic vitality. Open source is based upon a different set of values; academic values. These appeal to us as well. Sharing is good, duplication of work is wasteful. Two heads are better than one, and if organized properly, one hundred heads are considerably better than one. This study has brought to light some ways in which these two sets of values can be reconciled.

Government is Needed

This study began with the assumption that government action would be necessary if open source is to succeed in the short term. After considering the evidence uncovered in the course of this project, I still believe this to be true. The main reason for this is inertia, a factor that I believe open source evangelists like Eric Raymond overlook. The proprietary model has twenty years of inertia as the dominant way of doing business; it has come to be taken for granted by both companies and consumers. Linux, and open source in general, has thirty years of inertia towards being designed for programmers and other savvy users, emphasizing power and flexibility over aesthetics and ease of use. Clearly, both of these are changing: businesses are increasingly willing to toss away the proprietary paradigm in the name of flexibility, and open source developers have redoubled their efforts to create friendly graphical interfaces for Linux on which to run other open source software.

Yet inertia will continue to be a problem for a long time. The above-mentioned assumptions are ingrained in the organizational structures of companies, in the distribution

infrastructure of software retail and mail-order firms, in the minds of users, and most importantly, in the huge pool of existing software installed in millions of computers around the world. Software does not wear out the way a physical object does; it needs replacement only when users perceive that it is no longer useful. The tendency of users to stick with what they have—that which is familiar and suits their needs, however imperfectly—rather than install an entirely new suite of software tools, is perhaps the most formidable obstacle facing open source.

This inertia may doom open source perpetually to the fringes of the software industry, or at least make its acceptance an extremely slow process If activists want the change to occur sooner—if the social benefits of open source are worth having now—then government has a role to play. This role is not to shape the future direction of the movement, nor to codify open source methods into law or channel them to particular purposes, but to give the movement a running start, blast through some of the inertia, and level the playing field for fair competition against proprietary software.

Cooperation is Vital

Another fact that became apparent over the course of this research is that, since the help of government is needed in the short term, no organization can effect the change all by itself. Coalitions will be vital to any multifaceted open source lobbying effort. Hackers provide the technical expertise, the ideological zeal, and proof of the method's effectiveness. Public interest groups bring expert skill in government lobbying, a widely distributed base of support, access to a network of like-minded and active social change organizations, and a commitment to the social benefits of open source, beyond its economic benefit. Finally, companies provide financial resources, marketing apparatus, and support from a pro-business Congress. All of these resources will be necessary for successful public policy action.

What these various groups do not seem to realize is that no matter who their targets are, whether they seek to persuade the business sector, government, or the public, whether they call it "marketing" or "grassroots organizing" or "lobbying," the techniques are similar, and thus they have a lot to learn from each other. With this learning comes the potential for presenting a united persuasive front, sending complimentary messages to all relevant targets. For all of them, the key is characterization: stating an idea in a way which makes people respond positively, matching it to things they approve of and things they already do.

This coalition will not be easy to create or maintain, because it must contain such radically different viewpoints. Staunchly libertarian hackers, who believe in assigning to government as small a role as possible in all areas of life, must ally with Ralph Nader affiliates whose efforts at using government as a tool to correct societal ills have made headlines for decades. Conflicts will undoubtedly arise and details squabbled over. Fortunately, some of these differences are in fact superficial: although hackers may object in principle to government involvement in the economy, the policies they are in fact willing to live with (procurement, and probably research grants, according to Eric Raymond) are the very ones that seem to have the best chance of succeeding. Open source has the potential to address a number of problems, both economic and social. If the interested groups can work together, each can achieve the solution it is looking for, and in a shorter time frame.

The Trial is a Window of Opportunity

The lawsuit against Microsoft presents open source with a significant period of opportunity. To quote William Browne, "The stars are aligned in the policy heavens." Thanks to extensive media coverage of the trial, public and congressional discourse about monopoly, antitrust, and alternative solutions are at their highest point in years. Now, while policymakers are willing to listen, is the time to present alternative remedies, including open source. It was growing resentment about Microsoft's practices that spurred

the re-emergence of open source as a mainstream idea, and advocates should use their image as heroes battling the evil Microsoft, as David confronting Goliath, to the fullest advantage. Having an identifiable enemy gives one's message a certain legitimacy. Even although open source is more than just an anti-monopoly policy, making use of that characterization is a good move right now.

A Final Word

As Cox points out, the only solution to the software industry's complexity crisis is a paradigm shift, and paradigm shifts are never easy (Cox 1996, p. 50). Open source is undoubtedly such a paradigm shift, because it challenges fundamental beliefs and value systems. These phenomena, warns Cox, are chaotic, disruptive, and perceived by many as unnecessary. They are slow, and invariably cause some to lose profits in the short run. They tend to overthrow entrenched establishments. He uses the example of Copernicus destroying an entire worldview by placing the sun at the center of the universe rather than the earth. To this I add my own analogy: suppose Copernicus could have convinced the church to help promote his views? The social upheaval could have been mitigated. Although the paradigm shift of the open source revolution will not be easy, government can ease the transition.

Sources

Books and Articles

Areeda, Phillip (1992). "Antitrust Law as Industrial Policy: Should Judges and Juries Make It?" in Antirtust, Innovation, and Competitiveness, Thomas M. Jorde and David J. Teece, eds. New York: Oxford University Press. pp. 47-81.

Band, Jonathan. (1995). "Competing definitions of 'openness' on the NII." in *Standards Policy for Information Infrastructure*. Brian Kahin and Janet Abbate, eds. Cambridge: MIT Press. p. 351.

Bradner, Scott. (1998). "The Internet Engineering Task Force." in *Open Sources: Voices from the Open Source Revolution*. Chris DiBona, Sam Ockman, and Mark Stone, eds. Sebastopol, CA: O'Reilly & Associates. pp. 47-52.

Branscomb, Lewis, and Brian Kahin (1995). "Standards Processes and Objectives for the National Informatuion Infrastructure." in *Standards Policy for Information Infrastructure*. Brian Kahin and Janet Abbate, eds. Cambridge: MIT Press. pp. 3-31.

Brooks, Frederick P. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley.

Browne, William P. (1998). *Groups, Interests, and U.S. Public Policy*, Washington: Georgetown University Press.

Cassidy, John. (1998). "The Force of an Idea." *The New Yorker*. January 12, 1998. p. 32.

Commoner, Barry. (1979). *The Politics of Energy*. New York: Alfred A. Knopf.

Cox, Brad. (1996). *Superdistribution: Objects as Property on the Electronic Frontier*. Reading, MA: Addison-Wesley.

DiBona, Chris, Sam Ockman, and Mark Stone. (1998). Introduction and conclusion to *Open Sources: Voices from the Open Source Revolution.*. Sebastopol, CA: O'Reilly & Associates.

Dyson, Esther. (1998). *Release 2.1: A Design for Living in the Digital Age*. New York: Broadway Books.

Einstein, David. (1998). "The Penguin That Roared." *The San Francisco Chronicle*. September 8, 1998. p. B3.

Ellis, William. (1995). "Intellectual Property Rights and High Technology Standards." in *Standards Policy for Information Infrastructure*. Brian Kahin and Janet Abbate, eds. Cambridge: MIT Press. p. 450.

Farrell, Joseph. (1995). "Arguments for Weaker Intellectual Property Protection in Network Industries." in *Standards Policy for Information Infrastructure*. Brian Kahin and Janet Abbate, eds. Cambridge: MIT Press. pp. 368-377.

Federal Energy Administration. (1974). *Project Independence Blueprint Final Task Force Report: Solar Energy*. Government Document FE 1.18:So 4. ch. VII.

Gruenwald, Juliana. (1998). "Hoping to Fend Off Regulation, High-Tech Industry Steps Up Its Campaign Contributions." *CQ Weekly*. October 31, 1998. pp. 2958-2959.

Hafner, Katie, and Matthew Lyon. (1996).*Where Wizards Stay Up Late: The Origins of the Internet.* New York: Simon & Shuster.

Hall, Mark, and John Barry. (1990). *Sunburst: The Ascent of Sun Microsystems.* Chicago: Contemporary Books. pp. 103-105.

Hempel, Lamont. (1982). *The Politics of Sunshine: An Inquiry Into the Origin, Growth. and Ideological Character of the Solar Movement in America.* doctoral dissertation. Claremont, CA: Claremont Graduate School.

Hrebenar, Ronald, and Ruth K. Scott. (1982). *Interest Group Politics in America..* Englewood Cliffs, NJ: Prentice-Hall.

Kahn, Robert E. (1994). "The Role of Government in the Evolution of the Internet." *Communications of the ACM*. Vol. 37 No. 8. August 1994. pp. 15-19.

Leibovich, Mark. (1998). "The Spreading Grass-Roots Threat to Microsoft." *The Washington Post*. December 3,1998. pp. A16-17.

Munn, William G. (1999). *Constructing the Problem of Access to Information Technology*. doctoral dissertation. Claremont, CA: Claremont Graduate University.

New York Times Editorial. (1999). "Settling the Microsoft Case." *The New York Times.* April 4, Section 4; Page 10; Column 1.

Perens, Bruce. (1998). "The Open Source Definition." in*Open Sources: Voices from the Open Source Revolution*. Chris DiBona, Sam Ockman, and Mark Stone, eds. Sebastopol, CA: O'Reilly & Associates. pp.171-188.

Peters, B. Guy. (1996). *American Public Policy: Promise and Performance*. Chatham, NJ: Chatham House.

Raymond, Eric. (1998;1). "A Brief History of Hackerdom." in *Open Sources: Voices from the Open Source Revolution*. Chris DiBona, Sam Ockman, and Mark Stone, eds. Sebastopol, CA: O'Reilly & Associates. pp.19-29.

Raymond, Eric. (1998;2). "The Revenge of the Hackers." in *Open Sources: Voices from the Open Source Revolution..* Chris DiBona, Sam Ockman, and Mark Stone, eds. Sebastopol, CA: O'Reilly & Associates. pp.207-219.

Ripley, Randall, and Grace Franklin. (1986). *Policy Implementation and Bureaucracy.* Chicago: Dorsey Press. pp. 71-89.

Stallman, Richard. (1998). "The GNU Operating System and the Free Software Movement." from *Open Sources: Voices from the Open Source Revolution.* Chris DiBona, Sam Ockman, and Mark Stone, editors. Sebastopol, CA: O'Reilly & Associates. pp.53-70.

Shurmer, Mark. and Gary Lea. (1995). "Telecommunications Standardization and Intellectual Property Rights: A Fundamental Dilemma?" in *Standards Policy for Information Infrastructure*. Brian Kahin and Janet Abbate, eds. Cambridge: MIT Press. p.378.

Stone, Deborah. (1997). *Policy Paradox: The Art of Political Decision Making*. New York: W.W. Norton.

Tiemann, Michael. (1998). Future of Cygnus Solutions. from Open Sources: Voices from the Open Source Revolution. Chris DiBona, Sam Ockman, and Mark Stone, editors. O'Reilly & Associates. pp.71-89.

Vixie, Paul. (1998). "Software Engineering." from *Open Sources: Voices from the Open Source Revolution*. Chris DiBona, Sam Ockman, and Mark Stone, editors. O'Reilly & Associates. p.98.

Wilke, John R. (1998). "Sen. Hatch Issues Warning Microsoft May Be Building 'Proprietary Internet.'" *TheWall Street Journal.* February 6, 1998. Section B, Page 15, Column 1.

Worthington, Richard. (1984). "Renewable Energy and Politics: The Case of the Windfal Profits Tax." *Policy Studies Journal.* December 1984. pp. 365-375.

Yoffie, David B., and Michael A. Cusumano. (1999). "Judo Strategy: the Competitive Dynamics of Internet Time." *Harvard Business Review.* January-February 1999. pp. 71-81.

Electronic Sources

All Internet addresses mentioned in this section are current as of April 30, 1999. Dates, where given, refer to the document's date of publishing or of last modification.

Anonymous. (1999). "If Microsoft Built Cars." *Stokely Consulting web site.* <http://www.stokely.com/lighter.side/ms.cars.html>

Apache Group. (1999). "About the Apache HTTP Server Project." *Apache web site.* February 1999. <http://www.apache.org/ABOUT_APACHE.html>

EdLiNC. (1999). "About the E-Rate." *Education and Library Networks Coalition web site.* April 27, 1999. <http://198.5.6.209/erate/>

Free Software Foundation. (1998;1). "Categories of Free and Non-Free Software." *GNU's Not Unix—the GNU Project and the Free Software Foundation.* December 17, 1998. <http://www.fsf.org/philosophy/categories.html>

Free Software Foundation. (1998;2). "The GNU General Public License." *GNU's Not Unix—the GNU Project and the Free Software Foundation.* Version 2. February 16, 1998. <http://www.fsf.org/copyleft/gpl.html>

Liebowitz, S. J., and Stephen E. Margolis. (1995;1). "Are Network Externalities A New Source Of Market Failure?" *Research in Law and Economics.* Vol 17. pp. 1-22. Available: <http://wwwpub.utdallas.edu/~liebowit/netwextn.html>

Liebowitz, S. J., and Stephen E. Margolis. (1995;2). "Chicken Little Comes Home to Roost: A Misplaced and Flawed Economic Theory Bedevils Microsoft." *Upside Magazine*. September 1995. Available: <http://wwwpub.utdallas.edu/~liebowit/upside.html>

McNamara, Paul. (1998). "Linux cynics." *NWFusion Magazine*. November 6, 1998. <http://www.nwfusion.com/news/1109linux.html>

Microsoft Corporation. (1995). "Microsoft End User License Agreement, Internet Explorer." file included in *Internet Explorer* software package.

Microsoft Corporation. (1998). "Economic Experts Challenge Underlying Assumptions in Government's Case." *Microsoft press release*. October 16, 1998. <http://www.microsoft.com/presspass/features/1998/10-16econ.htm>

National Science Foundation. (1999). "President Asks Almost $4 Billion for NSF's Fiscal Year 2000 Budget." *NSF News*. February 1, 1999. <http://www.nsf.gov/search97cgi/vtopic?action=View&VdkVgwKey=http%3A%2F%2Fwww%2Ensf%2Egov%2Fod%2Flpa%2Fnews%2Fpress%2F99%2Fpr995%2Ehtm&DocOffset=4&DocsFound=5&QueryZip=IT2&Collection=NSFweb&SearchUrl=,>

NetAction. (1996;1). "Welcome to NetAction." *NetAction web site*. <www.netaction.org>

NetAction. (1996;2). "Don't Be Soft On Microsoft." *NetAction web site*. <http://www.netaction.org/msoft/index.html>

Newman, Nathan. (1997). "From Microsoft Word to Microsoft World: How Microsoft is Building a Global Monopoly." *NetAction web site*. <http://www.netaction.org/msoft/world/>

Newman, Nathan. (1999). "The Origins and Future of Open Source Software." *NetAction web site*. <http://www.netaction.org/opensrc/future/>

Open Source Initiative. (1997). "The Open Source Definition." *Open Source Initiative web site*. June, 1997. <www.opensource.org/osd.html>

Open Source Initiative. (1998;1). Home page. *Open Source Initiative web site*. Version 1.4. February 24, 1998. <www.opensource.org>

Open Source Initiative. (1998;2). "The Case for Open Source: Hackers' Version." *Open Source Initiative web site*. <http://www.opensource.org/for-hackers.html>

Raymond, Eric. (1997). "The Cathedral and the Bazaar." *Eric Raymond web site*. June 1997. <http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html>

Raymond, Eric. (1999). "How To Become a Hacker." *Eric Raymond web site*. March 1999. <http://www.tuxedo.org/~esr/faqs/hacker-howto.html>

Slabodkin, Gregory. (1998). "Software glitches leave Navy Smart Ship dead in the water." *Government News*. July 13, 1998. Available: <http://www.gcn.com/gcn/1998/July13/cov2.htm>

U.S. Department of Commerce. (1993). "The National Information Infrastructure: An Agenda For Action." <http://metalab.unc.edu/nii/toc.html>

U.S. Department of Justice. (1998). "United States of America v. Microsoft Corporation, Civil Action No. 98-1232 (Antitrust) COMPLAINT." *Department of Justice web site*. May 14, 1998. <http://www.usdoj.gov/atr/cases/f1700/1763.htm>

Valloppillil, Vinod. (1998). "Open source: a New(?) Development Methodology." Available at *the Open Source Initiative web site* as "Halloween I" with annotations by Eric Raymond. v1.00. August 11, 1998. <http://www.opensource.org/halloween1.html>

Zawinski, Jamie. (1999). "Netscape and aol." *Jamie Zawinski web site*. March 31, 1999. <http://www.jwz.org/gruntle/aol.html>

Primary Sources

Clark, Judi. *Personal interview*. October 9, 1998. San Francisco, CA.

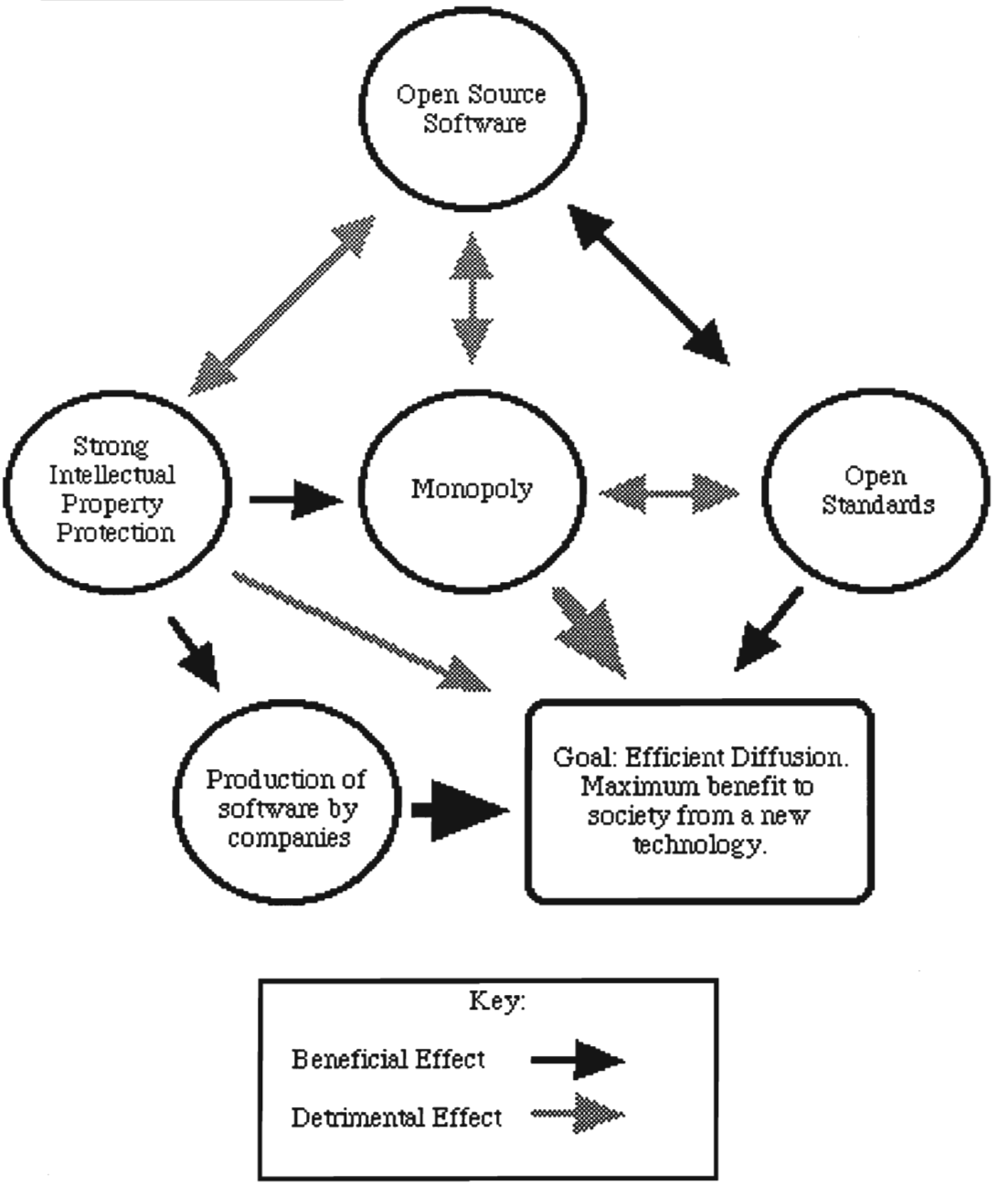
Hughes, Phil. *Telephone conversation. October 29, 1998.*

Krause, Audrie. *Personal interview*. October 9, 1998. San Francisco, CA.

Raymond, Eric. *Telephone conversation.* November 17, 1998.

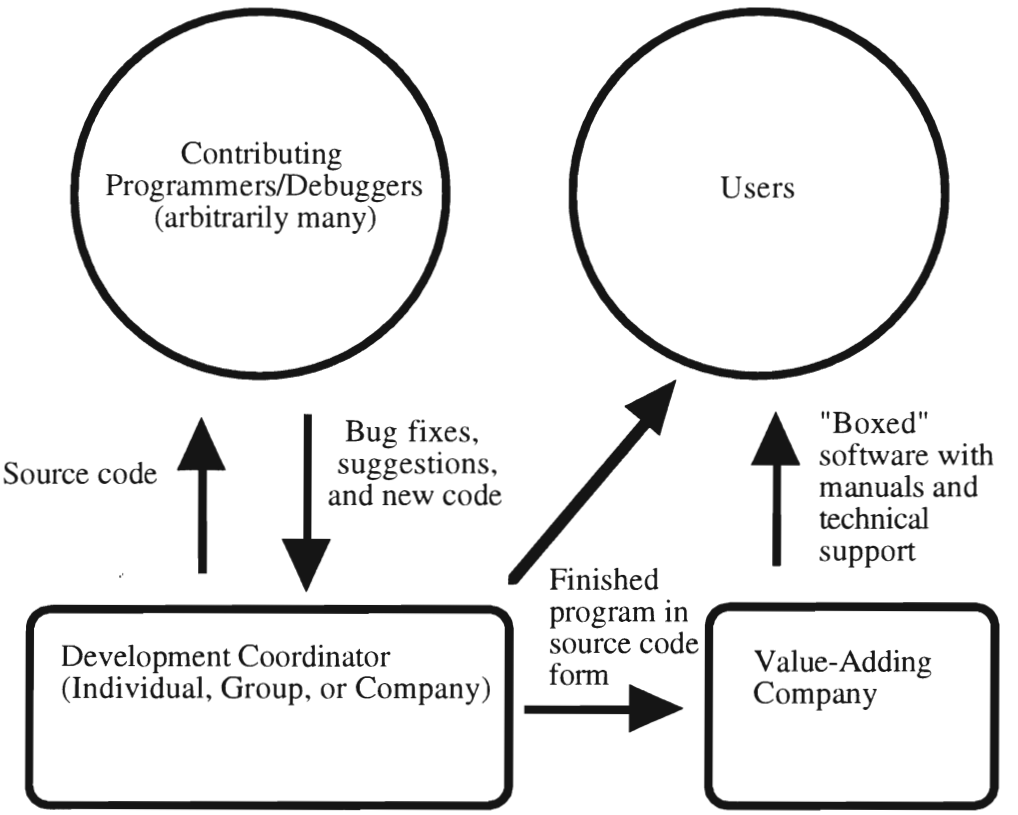
Appendix
Figures

Figure One:
Relationships between Legal and Economic Factors



This figure sums up the relationships discussed in Chapter 2. The rectangular box represents the overall goal of the software industry and the circles represent economic and social factors that contribute to or inhibit this goal. Note that although strong intellectual property protection can lead to monopoly, it is also an incentive for proprietary firms to produce more software. Note also that all of the arrows pointing to open source are two-way: open source both affects and is affected by the other factors.

Figure Two:
The Open Source Process



This figure depicts the open source process. The arrows on the left-hand side represent the development part of the process, in which some coordinating group makes its work-in-progress available to outside programmers, who contribute bug fixes and new code back to the coordinator. The final product, in source code form, is made available on the Internet for anyone to download. A value-adding company, if one exists, takes this source code and creates a "boxed" distribution with manuals and technical support. In reality, the boundaries between the groups are not as well-defined as this figure depicts them. Contributing programmers may move in and out of the coordinating group, users may become contributors, and the coordinator and value-adding company may be one and the same.