Research Publication No. 1999-02
4/1999

The Power of Openness

Why Citizens, Education, Government and Business Should Care About the Coming Revolution in Open Source Code Software

David Bollier

THE POWER OF OPENNESS

Why Citizens, Education, Government and Business Should Care About the Coming Revolution in Open Source Code Software

A Critique and a Proposal for the H20 Project

BY DAVID BOLLIER

http://www.opencode.org

Introduction

As software and networking technologies rapidly insinuate themselves into the deepest reaches of American commerce, culture and governance, the architecture of our democratic society is being transformed. One lesson that is becoming clearer is that the design of hardware and software and the governance of the Internet matters. These issues can profoundly affect competition and innovation in markets, the ability of universities, libraries and nonprofits to pursue their missions, and the control that individuals can exercise over their lives.

Within the past year, a number of forces have converged to suggest the socially constructive potential of software whose design code can be freely accessed and modified by computer users. It is a complex story that is still unfolding and known chiefly in computing/Internet circles. As we will explain in Section I, a growing grassroots movement on a global scale is challenging proprietary models of software development by generating superior, more reliable software that is far cheaper and even free.

The implications are not just technical but economic, political and cultural. A new software movement based chiefly on free, open access to the source code of software, is showing its tremendous power to fortify user sovereignty in the computing/Internet marketplace. This movement represents one of the most novel, potentially powerful expressions of the consumer movement in a generation. Over time, if fully developed, the new models of software development could produce innovative, cost-efficient software at much less cost than today, a capacity that could particularly benefit the voluntary, academic, nonprofit and professional communities. It could also help check the excesses of a market-dominated culture by fortifying these "gift culture" communities while mitigating worrisome concentrations of corporate power in the software industry.

But the rich latencies of this Internet-facilitated phenomenon may never develop if a new kind of networking leadership does not coalesce to assert the important values that can only flourish in an environment of openness. The user community and many non-technical constituencies must begin to identify and advance their strategic interests in open code software. Such a mobilization of resources is especially needed because many segments of the software and computer industries seem committed to containing the expansion of open code software, but, for now, have not consolidated enough to develop a unified response.

In the meantime, fissures among the proponents of open source code software may complicate the forward momentum of this movement. One branch, the Free Software movement, sees distinct moral, social and civic value in the source code of software being legally available to anyone in perpetuity. Another branch, the Open Source movement, is more concerned with the technical superiority of open source code software and its intriguing commercial possibilities. Rather than privilege one or the other branch of the movement - whose strategic objectives both overlap and diverge -- this essay will refer to both collectively as "open code" (not "open source") unless one or the other is specifically intended. In the interest of inclusiveness, this text will also use the term "new software movement" to refer to new non-proprietary models of software development and alternative intellectual property regimes.[1]

To the layperson, it may not be immediately apparent why the new software movement holds rich promise. This proposal seeks to explain why its surprising emergence into the mainstream in 1998 is so significant and how, with timely and strategic assistance, open code software could evolve into a powerful new platform for the reinvigoration of the non-commercial civic sector in American society. By radically empowering computer users, it could help rejuvenate our democratic culture, improve education, catalyze a more consumer-responsive economy and ensure fairer, open governance of the Internet.

To reap the full potential of the open code revolution, we propose creating H20, an independent nonprofit organization to help foster the development and usage of the new software. We use the metaphor of water, H20, to emphasize that software is increasingly indispensable to life, at least an enfranchised civic, political, economic and social life. If software is to truly improve these domains, it must be capable of circulating freely so that it can nourish the fundamental values of an open society: democratic participation, social equity, equal opportunity and educational achievement, among others. Openness is a virtue in software development not just because it tends to produce superior product, but equally because it fortifies free market competition and democratic principles.

This essay explains why open code software is so important, especially for various non-technical constituencies, and how a new organization - H20 - could help promote new development and usage of open source code software. This document is deliberately aimed at the layperson as much as the computer sophisticate because a new and broader conversation must be started, one that considers the far-reaching implications of open code software for how we shall govern ourselves, improve education, foster innovation and economic growth, and protect the sovereign interests of citizens and consumers. This text, then, is intended for anyone interested in these realms as well as for influential leaders of the foundation community who could catalyze some powerful changes by fostering the development of open source code software.

The plan of this essay is as follows:

I. THE RICH PROMISE OF OPEN CODE SOFTWARE

The "deep architecture" of global society is increasingly predicated on software, computers and telecommunications. In diverse permutations, these technologies are changing the skeleton, sinews and central nervous system of economy, democracy, organizations and public life. The kinds of knowledge that can be created and shared, the ways that markets are structured, the functioning of democracy, the kinds of social connections and identities that we develop, the kinds of public cultural spaces and conversations that can flourish: all are affected by the ongoing revolution in computers and electronic communications, especially the Internet.

Since software lies at the heart of so much of our new societal architecture, it is important to understand what forces are shaping the structure and evolution of software design. Software resembles a genetic force; entire worlds are enfolded within its embryonic design. As a theoretical matter, software is highly malleable, even protean. In practice, commercial software companies generally manage the design and evolution of their products to enhance their competitive advantage, a calculus that may or may not serve users' best interests. Online commentator Tom Hull explains:

> [I]n the world we live in, production is highly organized and efficient and commands enormous financial resources and seductive powers of persuasion, while demand is fragmented, uninformed and powerless. While consumers can still kill a product that they have no desire for, they are nearly powerless to direct or even influence the detailed designs of those products. For software products, consumers can only choose among a given set of alternatives, which are extremely complex, dauntingly impenetrable, and generally designed more for the company's anticompetitive purposes than for the user's tasks. [2]

[I]n the world we live in, production is highly organized and efficient and commands enormous financial resources and seductive powers of persuasion, while demand is fragmented, uninformed and powerless. While consumers can still kill a product that they have no desire for, they are nearly powerless to direct or even influence the detailed designs of those products. For software products, consumers can only choose among a given set of alternatives, which are extremely complex, dauntingly impenetrable, and generally designed more for the company's anticompetitive purposes than for the user's tasks. [2]

The revelations of the current antitrust trial against Microsoft dramatically illustrate this point. The U.S. Department of Justice alleges that Microsoft has used its dominance of the computer operating system market to unfairly restrict consumer choice of other operating systems, browsers and other software applications. As the maker of more than 90% of all desktop computer operating systems, Microsoft has been able to charge higher prices for its software through restrictive licensing agreements with computer equipment makers, despite the availability of superior alternatives. Through a strategy described at the Microsoft antitrust trial as "embrace, extend and extinguish," the company has taken open standard protocols such as HTML (for web pages), Java (the cross-platform software), RealAudio (the Internet audio software) and QuickTime (multimedia software), and sought to sabotage them by introducing its own proprietary modifications as the de facto standard. Microsoft has also aggressively used a FUD strategy - "fear, uncertainty, doubt" - to scare would-be competitors from entering markets Microsoft wants to dominate. Such aggressive actions have chilled innovation in the software industry by forcing other developers to retreat to market niches where fair competition is possible.

While Microsoft may be a singular force in computing, its alleged tactics are simply an extreme case of what other software companies would likely use as well, if they could. Many software companies aspire to use ingenious proprietary design standards to create or dominate new market niches, "lock in" their technology, and preemptively limit competition and user choices. For the most part, the "seller side" is committed to determining what types of software innovations will be allowed to compete in a given market; what types of licensing arrangements shall govern users; what prices consumers will pay for products whose incremental costs of production are virtually nil; and what websites will be given preferential access or development by the primary Internet gatekeepers/portals.

It is important to understand that software design can be used strategically to shape and control markets and even national cultures. This is illustrated by PICS, the Platform for Internet Content Selection, a sophisticated HTML standard that allows private agencies, governments and corporations to rate and block online content.

Although "content-neutral" in design (it can filter out birdwatching literature as easily as Nazi Party diatribes), the PICS technology is an unparalleled tool for censorship on a global scale. The development of PICS by a small cadre of little-known technicians and business executives suggests how software can be used to "design" the character of markets, political life, and culture. The so-called Y2K or "Year 2000" problem also illustrates how even a small but structurally entrenched problem in software design (the truncated field that bedevils storage and retrieval of the year value for years that do not begin with 19xx) can affect everything from air traffic control and international currency trading to emergency services.

A. The Importance of Source Code in Software

Proprietary source code, protected by copyright law, plays a central role in enabling the "architectural design power" of software. Originally, copyright arose to protect and reward creative expression in printed media in order to assure its public dissemination. But copyright is often misused as a legal tool for withholding creative expression and controlling the terms of competition in a given market. The seller can prevent users from altering the functionality of a software product, for example, or limit its interoperability with other software and hardware. The seller can prevent users from customizing the product to suit their own needs. The seller can ignore known bugs and other product flaws, and in other ways force users to accept unwanted design features in a product.

Historically, software makers have sought to preserve these prerogatives (and thus their relative advantage over competitors and consumers) by vigorously protecting their "source code," the internal blueprint of a software product that is the basis for the "binary code," or binaries, the user-accessible manifestation of a software product that a computer executes. As long as the proprietary source code can be held as a trade secret protected through copyright law, sellers retain greater control than users over the terms of software design, quality control, pace of innovation and use.

However important proprietary control of source code may be in some respects, that control diminishes consumer sovereignty, accountability and choice. In many instances, it results in anticompetitive mischief. This is especially pernicious because, in rapidly changing markets with technically complex products, neither consumers nor antitrust regulators are truly equipped to challenge such marketplace abuses except in the most egregious instances (such as the Justice Department's action against Microsoft), and the ultimate remedies secured may be deficient in any case.

The new software movement offers a radically new basis of competition, innovation and consumer power in the marketplace. Propelled as if from nowhere by converging forces, the new software movement has been catapulted into the mainstream over the past year. This user-driven model being generated by a vast global community of computing irregulars not only rivals proprietary software in quality, reliability, flexibility and price; it effectively "belongs" to the user community and therefore offers a cheap, versatile and durable scaffolding for the self-development of all sorts of user communities.

Just as the cooperative movement has enabled workers and consumers to assert greater control over their economic and personal lives, so the new software movement opens up new frontiers of self-determination. It allows users to pole-vault over the cost inefficiencies, barriers to innovation, consumer manipulation and design rigidities that characterize many proprietary software markets. (Open code communities are able to bypass most of the traditional impediments to coops because the Internet allows incremental and ad hoc participation, participation without regard to geography, avoidance of overhead and management costs, and rapid feedback loops with consumers.)

For all its promise and remarkable resourcefulness, the new software movement faces many serious challenges if it is going to develop a more substantial influence in the marketplace, in life, in cyberspace. Before examining that issue in Section II below, it is important to understand some of the deep dynamics of the new software movement, what makes its software so robust and innovative, and how it could, with the kinds of assistance H20 proposes, enable myriad social, educational and civic benefits.

B. The Growth of the New Software Movement

In the early days of computing, a great deal of software development was a collaborative process carried out by academics and students. Software was seen as the shared product of a community, to which everyone freely contributed and benefited. In this respect, open code software shares the strength and resiliency of the scientific method and Jeffersonian democracy. All procedures and outcomes are subject to the scrutiny of all. Openness allows error to be more rapidly identified and corrected. Innovation and improvement can be more readily embraced. Openness builds accountability into the process of change.

The commercialization of computing in the 1970s and 1980s introduced a new dynamic to software development: a closed, proprietary process that mobilized expertise to develop innovations, bring them to the market, and reward private investors. Bill Gates' entrepreneurial passion was so great he was nearly expelled from Harvard for using publicly funded labs to create commercial software - a violation of the hacker ethic of sharing (and not privatizing) community knowledge. After Gates was required to put his code in the public domain, as free software, he quit Harvard and went on to found Microsoft. The high-tech entrepreneurialism that Gates exemplifies is now the stuff of American legend.

Yet lurking in the shadow of this mighty new industry over which Bill Gates reigns, the free software movement has quietly persisted and grown. Empowered by the Internet, the community of computer aficionados willing to develop, improve and freely share software among themselves has mushroomed. Even as Silicon Valley gives birth to dozens of startup ventures each year, driven in no small part by the connectivity of the Internet, so the hacker community has grown in sophistication, breadth and international scope. It has generated hundreds of top-quality software programs, many of which have become critical operating operating components of the Internet.[3]

What most distinguishes this free software from the off-the-shelf proprietary products is the openness of its source code - and the user's freedom to use and distribute the software in whatever ways desired. Anyone with the expertise can "look under the hood" of the software and modify the engine, change the carburetor or install turbo-chargers. Inelegant designs can be changed, noisome bugs can be fixed (or introduced). Sellers cannot coerce users into buying "bloatware" (overblown, inefficient software packages with gratuitous features), Windows-compatible applications or gratuitous software upgrades contrived through planned obsolescence. Nor are constant upgrades in computer hardware (such as the latest, high-speed Intel chips) required.

In short, open code software allows users to assert much greater control over their computing environment. This is not a matter of customizing a spell-checker; users with access to source code can change basic functionalities of their software and reap significant new efficiencies, often for free. By enabling skilled users to customize software to suit their special needs, open code software represents an entirely new kind of media empowerment. It potentially allows academic specialties, libraries, distance educators, civic organizations, business enterprises and others to develop their own innovative vehicles for sharing and elaborating a common body of knowledge. They do not have to adapt to the design structures, constraints on innovation and licensing/price schemes imposed by proprietary vendors. In this sense, open code software is not just a product, but a new kind of knowledge- and community-building infrastructure.

The benefits that users reap from open code software - customization, innovation, education, security, efficiency, reliability, cost savings - are actually "symptoms" of their collective empowerment as users. By banding together to assert their common interests, open code software users acquire an entirely new dimension of power. They are not merely consumers picking and choosing from the products sellers may choose (or decline) to offer; they are co-producers in the creation of specific software to serve their distinctive needs. The power wielded by the open-code user networks is vastly greater than that which propelled public interest initiatives in other media -- universal service in telephony, weak rights of citizen access in television and radio, a chronically beleaguered public broadcasting system; and local access channels for cable television. All these initiatives have required political interventions, court litigation or significant government spending, which means that have been constantly vulnerable. By contrast, the open code software movement is a sovereign, expansive "political" force in its own right.

C. The Alternative Economics of Open Code Software Development

Although critically aided by the Internet, the power of the new software movement stems from the "gift culture" that lies at the heart of the open code development model. The Internet is a quintessential example of a gift culture. People are willing to make all sorts of useful information available for free, in defiance of orthodox economic "rules" that claim such voluntary behavior can occur only with financial incentives. The Internet is so robust precisely because people are giving of themselves without demanding a specific contractual payback. This is the very essence of community and civility. People are willing to enter into gift economies because they trust that they will at some point share in the "wealth" that the community freely passes among itself - much as an academic community freely shares its knowledge among its members and disdains those who seek to financially profit from the community's shared body of knowledge. The key point is not that the information or software is economically free - or "free beer," as the jibe goes - but that it is freely accessible and amenable to modification by anyone interested in doing so. (There are, of course, other reasons why people contribute software and open code to the public over the Internet. Software companies may want free publicity or a greater influence over user "mindshare"; a software developer may want to showcase his programming prowess; an academic community may wish to collectively share its resources.)

The phenomenon of leveraging the moral commitments of a community of people to achieve economic development and collective improvement is not exotic; it's simply denigrated by mainstream economic theory. Micro-lenders such as Grameen Bank in Bangladesh and South Shore Bank in Chicago have proven that they can make safe, profitable loans to inner-city entrepreneurs and collateral-poor mom-and-pop ventures that conventional banks find risky.[4] How? By realizing how trust, moral commitment to one's peers and a concern for reputation are not only economically valuable but often more powerful and motivational than economic incentives alone. Similarly, the "Time Dollars" program has also shown how moral and social values can be harnessed to achieve "economic" goals. People can earn "time dollars" (one hour, one "time dollar") to babysit, rake leaves or provide legal assistance for other members of the community; the credits can then be used to "buy" services from other participants in the network. With no exchange of money, a considerable amount of economically valuable activity is performed - work that conventional markets are often incapable of eliciting.

So it is with open code software, as facilitated by the Internet. Many of the crucibles that incubate open code software are "gift economies," whose economic development is tightly linked with community-building. By the lights of mainstream economics, the moral and community dynamics that help generate open code software simply do not make sense. But in real life, forget the theories: open code software development works. But however formidable its creative capabilities, open code software will be developed only in those domains where its value is recognized. And many cash-strapped enterprises within academia, the nonprofit world and community organizations - as well as many businesses -- do not yet recognize the remarkable tools they could develop. The leadership and education that H20 proposes could make a significant difference.

Richard Stallman, a programmer at the MIT Artificial Intelligence Laboratory in the 1970s, was one of the first to recognize the tensile strength of what he called "free software," in which "free" refers to the freedom to change the source code, not free in price.[5] For both personal and philosophical reasons, Stallman has strenuously objected to proprietary software because its copyright licensing restrictions limit his personal freedom to share programs with his friends and co-workers, to change the programs as he wishes, and to distribute improved versions that will benefit the larger community. He considers copyright control of software to be philosophically repugnant because it cripples the very advantages of digital technology -- its flexibility and ease of reproduction and sharing -- and encourages personally intrusive kinds of enforcement. As his peers trooped off to make fortunes in the emerging computer software industry in the early 1980s, Stallman instead founded the Free Software Foundation. The visionary project (which earned Stallman a MacArthur "genius" award grant) is dedicated to developing and promoting free software.[6]

D. The GNU Project and the GPL

Two innovations developed by Stallman and his colleagues have special relevance to the new software movement. First, Stallman in 1983 launched the GNU Project, a collaborative endeavor to develop a free Unix-like operating system (free in price as well as open source code). ("GNU" is an acronym for "GNU's Not Unix," a recursive pun; Unix, of course, is the open standards-based operating system developed by Bell Labs in the 1970s. Unix was the original operating system of the Internet and of governments and academic institutions, a role that it continues to play to this day.) A primary motive behind the ambitious GNU project, Stallman explained, was to "bring back the cooperative spirit that prevailed in the computing community in earlier days -- to make cooperation possible again by removing the obstacles to cooperation imposed by owners of proprietary software." Ultimately, Stallman wants to supplant proprietary operating systems by making them free, open-source code commodities. By the 1990s, the GNU Project had found or written all the major components of an operating system except one, the kernel.

Fortuitously, a free kernel developed by a Finnish student, Linus Torvalds, and greatly improved by a large self-organized network of volunteers, appeared on the Internet in 1991.[7] This kernel, when combined with the Unix utilities developed or collected by the Free Software Foundation over the past twenty years, "was like setting a match to dry leaves, creating an entirely new and completely open operating system overnight," according to one account. The conjoining of the GNU system with the so-called Linux kernel (Linus + Unix = Linux) has largely fulfilled Stallman's vision of a well-designed, free Unix-like operating system that can run on diverse hardware platforms. There are other free operating systems that command great respect among software developers, yet none has achieved the huge popularity that GNU/Linux has achieved in recent years. It has been a "stealth" operating system because technophiles in major companies often secretly used the software, knowing that their bosses would disapprove of the use of a "free" operating system of unknown provenance in mission-critical applications. Now that the reliability, versatility and price of GNU/Linux has received mainstream validation, a juggernaut has been unleashed. It has become the fastest-growing flavor of Unix and a preferred operating system in such diverse organizations as Boeing, Northern Telecom and NASA. It is also noteworthy that many enterprises and universities in Mexico, China, France, Australia and eastern Europe - fed up with paying expensive licensing fees for less reliable, versatile proprietary products -- are adopting GNU/Linux.

GNU/Linux usage received its biggest boost in 1998 when the press discovered Linus Torvalds, making him the iconic underdog of the computer world. Meanwhile, over the past year, one major software developer after another - IBM, Oracle, Corel, Inprise, Informix, others -- has announced they will port software applications to run on GNU/Linux. A number of vendors -- Red Hat, Caldera, SuSE, Pacific Hi-Tech - have begun to sell GNU/Linux along with documentation and technical support, attracting significant investments from the likes of Intel and Oracle. It is estimated that more than seven million people now use GNU/Linux as their operating system.

GNU/Linux might never have emerged but for Stallman's second innovation: the GNU General Public License (GPL), sometimes known as "copyleft." Stallman astutely realized that simply putting free software into the public domain was not enough, because anyone could make minor changes in a program and then copyright it, converting it back into a proprietary product. Without some legal vehicle, the benefits of free software could be privatized and withheld from the community of users. So Stallman developed the GPL, which is essentially copyright protection with special contractual terms. "To copyleft a program," writes Stallman, "first we copyright it; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify and redistribute the program's code or any program derived from it, but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable. Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom."

The GPL offers the greatest legal assurance that open source code will remain free and available to all, and that no company will appropriate the property for itself. But there are other copyright licensing variations. The BSD-style licenses -- based on the Berkeley Standard Distribution of Unix -- gives software users the option of creating derivative versions that can be copyrighted and made proprietary -- without revealing the source code. When Netscape released the source code of its Communicator browser in early 1998, it issued a Mozilla Public License, or MPL, which steers between the terms of the GPL and BSD licenses (private derivative versions can be made but any changes to source code covered by the MPL must be made freely available on the Internet).
Licenses that allow the privatization of the public pool of source code - thus creating a new proprietary line of development, often at the expense of the public pool -- is called "code forking." Stallman and other programmers are adamant about the principles of the GPL because code-forking vitiates the momentum of free software development. "Some in the open source community resent third parties taking from the public pool of software without contributing," writes publisher Tim O'Reilly in a major review of the open source movement. "In economies, this is called the free-rider problem. But despite the lack of a mandate, voluntary cooperation abounds."[8]

This philosophical divide is sometimes a contentious one, aggravated by the growing interest of technology companies in open code development models. It remains unclear if the powers of an open code "gift community" can be harnessed by software companies that answer ultimately to investors, not that community.

E. The Cathedral and the Bazaar

Although Stallman's GNU Project and Linux are two sentinel developments in the new software movement, they by no means define it. They are simply two of the largest manifestations of a more significant development, the open, Internet-facilitated process of software development. The idea that thousands of computer volunteers working together through the Internet might actually produce a sophisticated operating system that could out-perform proprietary software seemed ludicrous until GNU/Linux began to emerge. Open source theorist Eric Raymond explained the anomaly and thereby gave it wider credibility through a seminal online essay, "The Cathedral and the Bazaar," which first appeared in April 1997.[9] Raymond writes:

> Linux overturned much of what I thought I knew. I had been preaching the Unix gospel of small tools, rapid prototyping and evolutionary programming for years. But I also believed there was a certain critical complexity above which a more centralized, a priori approach was required. I believed that the most important software (operating systems and really large tools like Emacs) needed to be built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time.
>
> Linus Torvald's style of development -- release early and often, delegate everything you can, be open to the point of promiscuity -- came as a surprise. No quiet, reverent cathedral-building here -- rather the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from anyone) out of which a coherent and stable system could seemingly emerge only by a succession of miracles.

Raymond's great contribution was to explain in specific detail how and why a motley assemblage of thousands of hackers working for free on their own time ("the bazaar") could produce better software programs than the expensive professional talent amassed by Microsoft and other software companies ("the cathedral"). The answer has much to do with tapping into the decentralized intelligence of a huge pool of personally motivated programmers via the Internet. Others have observed that the cost of developing any software that is fundamentally better than current software will require more resources than any single corporation can afford - but that global networks of the best programmers motivated by prestige and peer recognition can generate vastly superior software, as GNU/Linux demonstrates.[10]

Some of Raymond's trenchant observations:

> -Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
> -Every good work of software starts by scratching a developer's personal itch....The best hacks start out as personal solutions to the author's everyday problems, and spread because the problem turns out to be typical for a large class of users.
> -Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
> -Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone. Or less formally, "Given enough eyeballs, all bugs are shallow."

The analysis of the "bazaar" model of software development was so compelling that Netscape, under siege by Microsoft's Internet Explorer, decided in January 1998 to release the source code of its Communicator browser. Netscape's move was a radical notion at the time, motivated at least as much by business calculation as moral conviction. Within months, many other software companies began to recognize the power and technical superiority of open code software -- as well as the niche profit opportunities in a marketplace dominated by Microsoft. Such major players as Oracle, Corel and Informix announced that they would be porting their products to GNU/Linux. Others, such as Sun, have made the source code of their programs available in some fashion or another, but under modified licensing rules that seek to maintain some measure of proprietary control. It has been especially significant that IBM has announced its support of GNU/Linux and the Apache Group (the open code community that has made Apache the dominant web server software) and that it will ship a free, no-frills version of its database program, DB2, for GNU/Linux in spring 1999.

The growing interest of software vendors and even chip-makers like Intel in open code software promises to trigger significant upheaval in the computing/software industries as the proprietary world seeks out new ways to exploit, domesticate or co-exist with open code software. It is unclear how GNU/Linux and other open-code software will fare if proprietary developers actively seek to arrest or control its growth. On the other hand, Red Hat, Caldera, SuSE and many other companies see novel business models based on a growing and robust open code software "market."

In any case, open code software clearly represents a serious alternative to the traditional proprietary model of software development. Its appeal is not just that it is cheaper, more versatile, reliable and customizable software. Open code software represents a structural shift of power from sellers to users, and in that sense is one of the most liberating tools of media empowerment that individual citizens and the civic sector might ever imagine. Open code re-positions the terms of competition to a matrix of quality and utility, and diminishes the advantages to be gained from manipulations of the distribution apparatus, marketing schemes, restrictive licensing terms, and bundling deals with hardware makers.

Open code software is also inherently more suited to educational environments because its inner logic - the source code - can be directly manipulated by students. With its inner parts visible, users can choose to learn how the software works, and then share and develop that knowledge. Proprietary software, by contrast, is inherently "unknowable" because its inner architecture is a trade secret.

The future of the open code software movement may hinge upon what new terms of engagement emerge between the proprietary world and open code software developers. Can the integrity of the free software vision and the vitality of its online communities be maintained as the movement expands and perhaps morphs into a more diverse open source movement? Section II explores the obstacles that must be addressed, followed by an agenda that H20 proposes for fostering the continued growth of the new software movement.

II. FORCES THAT MIGHT DERAIL THE OPEN CODE VISION

Open code software just may be the elusive "killer app" that the computer business has sought for so long: a powerful, novel software that "changes everything." Yes, its actual presence among average computer users is still quite limited. But as the poet Mary Oliver might put it, "It's not the size, it's the surge." GNU/Linux is the only non-Microsoft operating system showing growth in market share - 212% growth in market share in 1998; it has 17.2% of the market for server operating systems, a nearly three-fold gain over the previous year.[11]

Apache, Perl and many other open code software programs are being embraced by major proprietary vendors as cornerstones for future software development. Even Microsoft concedes in internal strategic analyses leaked to Eric Raymond -- the "Halloween Memos" -- that GNU/Linux and other open code software "provide very dramatic evidence....that commercial quality can be achieved/exceeded by open source software projects."[12] All of this suggests that the new software movement stands at a new threshold. It has a sovereign vision, a superior product, a burgeoning cadre of supporters, growing investment, fresh attention within technology circles and the general press, and a hardy development process of proven effectiveness.

But can this movement sprout wings and fly? Open code software faces some formidable challenges if it is to grow and become a broader consumer phenomenon. Among the barriers it faces are the lack of popularly accessible documentation and technical support; the lack of a clear, well-known brand identity and marketing support; a susceptibility to code-forking that can vitiate the development process; the shortage of enterprise-oriented development tools; development uncertainties that make proprietary vendors queasy about porting applications to open-code operating systems; and problems with integrated, end-to-end ease of use. These issues raise questions about the ultimate strength and durability of the open code movement, especially if proprietary vendors seek to domesticate or divert its software in order to protect their existing products and revenue flow.

This section outlines some of the challenges that proponents of open code software may face, both affirmatively in expanding the quality and usage of their products as well as defensively in neutralizing monkey-wrenching by competitors. It is unclear which of the challenges enumerated below will actually materialize and which opportunities should have the highest priority. But if the new software movement is to flourish, these issues certainly deserve further inquiry and discussion.

A. Can the Integrity of the Open Code Vision be Maintained?

The dynamic tensions between the proprietary "cathedral" models of software development and the open code model are likely to intensify in coming months and years. Indeed, this very polarity may become more complicated as new proprietary/open code business models are developed. A key question is whether the open code movement can retain the integrity of its vision and community-driven development process as the proprietary world comes calling. Will established companies see open code software as a threat or promise, or both? There is no question that Microsoft sees open code software as a threat to its franchise. As the Halloween I memo astutely notes, the most significant threat is the open software development process: "The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization effort appears to scale." Halloween I concedes that open code software "poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS [open source software] has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat."[13]

Apart from such outside threats, however, there are questions about whether the open code community will retain its community vitality and mutual commitment as its popularity (and opportunities for entrepreneurial spinoffs) grows. Also, can open code communities think and act strategically? Vinod Valloppillil, the author of Halloween I, shrewdly points out:

> A very sublime problem which will affect full scale consumer adoption of OSS projects is the lack of strategic direction in the OSS development cycle. While incremental improvement of the current bag of features in an OSS is very credible, future features have no organizational commitment to guarantee their development.

It is an open question whether "organizational commitment" is needed to stabilize and guide open code software development. Given the radically decentralized nature of the new software movement, however, it could benefit from a "loose leadership" that helps convene relevant players, focus discussion, scout out the frontiers and develop strategic analyses. Such efforts are important if the movement is going to respond to the inevitable threats that the proprietary software world will contrive. There may be some lessons to be learned from the 1960s, in which leader-less political initiatives were sometimes out-maneuvered by the more strategically organized, resource-rich opposition. Without losing the moral legitimacy and political strength of its grassroots base, the new software movement needs to evolve new governance/leadership structures to deliberate and advance its interests.

At the same time, the open code vision will require a broader, more ecumenical engagement with computing professionals, especially within corporate MIS departments, and non-technical constituencies such as education, government, nonprofits and the general user. Maintaining a core vision while popularizing it is a signal challenge.

B. Can the Open Code Community Overcome Intellectual Property Snares?

Many open code advocates correctly worry that software companies will create copyrighted derivatives that will privatize the shared intellectual capital of the open code community - the very scenario that inspired Richard Stallman to create the GNU General Public License. This has happened, for example, with SendMail, the open-code mail transport software whose original author, Eric Allman, has turned it into a proprietary product (while promising to improve and support the open-code version). This trend may accelerate as more programmers decide to become entrepreneurs feeding off of open code networks.

While such adaptations of open code programs may seem relatively benign, similar tactics by a Microsoft using "embrace, extend and extinguish" tactics could be disastrous. Microsoft speaks of trying to "fold extended functionality into commodity protocols/services and create new protocols." What this means, Eric Raymond explains, is "introducing nonstandard extensions (or entire alternative protocols) which are then saturation-marketed as standards, even though they're closed, undocumented or just specified enough to create an illusion of openness." This, of course, is precisely the Microsoft strategy that a federal court enjoined recently in Sun's litigation. It charged Microsoft with "standards pollution" of Java, which is intended to be a "write once, run anywhere" software - a "virtual machine" that can run on a variety of operating systems.[14]

Clearly the GPL offers the best protection for expansion of open code software. But that may not be enough. Some distribution vendors combine non-GPL software with the GNU/Linux kernel and distribute them, introducing new copyright quandaries. Also Microsoft may be able to use its market power to prevent open code software from reaching a wider base of consumers. It could prohibit OEMs [original equipment manufacturers] from pre-installing GNU/Linux, for example, as it appears to be doing with Dell in Europe.[15] Even though Microsoft has recently allowed computer makers to take new liberties in configurations of their opening screens (allegedly in response to the DOJ lawsuit), Microsoft's underlying contract terms with OEMs have not changed.

Nor are any PC makers accepting a challenge by the chairman of Be Inc. to pre-load either BeOS (his company's operating system) or GNU/Linux for free - which would have obvious value to consumers.[16] None will do so, contends Jean-Louis Gassée, because it would violate Microsoft licensing provisions, which he says prohibit OEMs from modifying the boot manager to display non-Microsoft operating systems or allowing the display of "unapproved" icons on the desktop screen. "As an industry insider gently explained to me," said Gassée, "Microsoft abides by a very simple principle: No cracks in the wall. Otherwise, water will seep in and sooner or later the masonry will crumble."[17] If Microsoft's licensing terms could be resisted with impunity by OEMs, on the other hand, it could allow real competition among operating systems. It would also vastly expand the potential user base for open code software and moderate the prices for Microsoft products.

There are also questions about hybrid corporate-sponsored "noospheres" (discrete communities of open code programmers)[18] such as Mozilla, the Netscape-sponsored group of programmers dedicated to extending and customizing Netscape's browser. Now that Netscape is being bought by AOL, there are fears within the Mozilla community that AOL-Netscape could abandon Mozilla or push its work into an AOL-owned proprietary tree. AOL-Netscape, for its part, have their own fears of legal liability if the open-code process were to introduce patented code into the browser.[19] Similarly, Sun is trying to kick-start a community of open code developers for its Solaris software using a quasi-open code license which it calls a Community Source License.[20] The long-term viability of these experiments remains an open question.

C. Can Open Standards and Interoperability be Assured?

Beyond desktop and enterprise computing, the new software movement is committed to an open frame-work for the Internet. Fundamentally, this means open standards and interoperability among different hardware and software platforms. Open standards means that the standard-setting process is open to anyone; that techni-cal specifications are published and freely available, and that no single vendor can control or manipulate the evolution of the standard. Ideally, there are certification tests to assure that an open technology actually meets the specifications.

The Halloween I memo suggests a chilling scenario by which Microsoft would subvert the commodity standards and infrastructure that is the foundation of the Internet by introducing proprietary protocols. The author suggests that Microsoft "de-commodize protocols and applications" by "extending these protocols and develop-ing new protocols." This would undermine the core advantage of open code software, its ability to integrate diverse hardware and software through common commodity protocols. This is why Microsoft finds open code software in general and GNU/Linux in particular so alarming: it threatens to subvert its proprietary integration of systems - and thence its monopoly rents - with open, competitive integration based on user-determined stan-dards of quality and utility. This is why Microsoft quietly changed its protocols to render Samba, an excellent open code communications application, incompatible with Microsoft's Windows NT and Windows 95/98 soft-ware.[21] This also explains why Microsoft is hostile to the IETF, the Internet Engineering Task Force, which has historically resisted the special pleadings of any single vendor in generating technical standards for the Internet.

When AT&T and the Bell Companies controlled the technical standards for telephonic network, no one else could compete. But once standardized interfaces were required to ensure interoperability, competition was possible - and innovation in all sorts of value-added products and services materialized. So it must be with the Internet, now and in the future. It is vital to ensure that the computer box, servers and switching protocols are interoperable, or else competition, innovation and their myriad benefits will be thwarted. The open code move-ment represents a powerful but under-mobilized constituency for assuring open standards and interoperability on the Internet.

D. Can Open Code Software Build a Brand Identity?

In our business-dominated culture, it is an inexorable reality that people respond to brands - an image and aura that distinguishes Brand x from Coca-Cola, McDonalds, General Motors and Microsoft. This may pose problems in expanding usage of open code software, because it has none of the contrived mystifications that come from advertising. It doesn't have an image, except perhaps its anti-image as a Birkenstock, counterculture artifact. While image may not matter much for techno-sophisticates, it could matter for the average consumer and even for the middle-managers of corporate MIS departments. Brand names are a form of cultural credibility. Even though that credibility may be unwarranted or unjustifiably expensive, it is, for now, a price of entry to the mainstream of American commerce and culture.

If it is going to expand beyond its grassroot niches and become an established player in mainstream computing, open code software may need to bring its image into focus and "manage" it with some strategic sophistication. Some argue that the modularity, technical superiority and increasing plug-and-play capacity of GNU/Linux may make traditional advertising gratuitous. This may prove true. On the other hand, the larger uni-verse of open code software could become a perennial also-ran - an uncredentialed, second-choice alternative, despite its superior merits - unless it has some minimal branding and marketing support.

One small but important step in this direction is OpenSource.Org, a certification system started by Eric Raymond and others, which gives a seal of approval to open code software that meets ten specific standards.

Under the organization's criteria, the GNU GPL, Berkeley Software Distribution, and Mozilla Public License, among others, meet the standards.[22] (A similar open-standards certification for hardware may also be necessary as more software functionality become embedded in hardware interfaces, threatening the open protocols that allow competition and innovation.)

What really needs to occur is a more coordinated cooperative association of powerful brands. This could resemble coop ad campaigns for an entire product category -- milk, eggs, books - financed by a group of vendors. On a parallel track, there could be an aggressive public education program launched through popular magazines, the general press and constituency-specific periodicals (e.g., higher education, nonprofit trade associations). There are also online vehicles, conferences and other vernacular venues through which the open code "message" could be popularized in highly credible ways.

It is unclear whether the various new software communities might wish to cooperate and "go legit" through advertising and marketing. Certainly such initiatives would need their support. But resistance could be self-destructive because ultimately, the "counter-cultural" identities of GNU/Linux, Apache and other open code projects are vulnerable to appropriation and caricature by competitors if there is no organized, strategic plan for "managing the identity" of the new software movement. As the Halloween memos frankly admit, this is a battle for "mindshare." Part of the success of open code products will require the savvy, pro-active cultivation of mainstream awareness and acceptance.

E. Can Open Code Software Be Made User-Friendly?

While GNU/Linux has achieved remarkable growth among the technically proficient, its future among the average computer user will require improvements in user-friendliness. The modular architecture of GNU/Linux, which makes perfect sense for the technically proficient, may prove troublesome for the mass-market consumer. Generally speaking, open code software has not undergone extensive testing or modification with non-techies to make them easier to use. They are, after all, products of the hacker community. Microsoft has built much of its empire on the seamless integration of a user-friendly operating system and applications: a value-added monopoly function, poorly achieved, for which it charges a significant premium.

The parallels with the pre-divestiture AT&T are striking. Before the 1984 AT&T consent decree, there was no competition; just a regulated monopoly. In the years after that watershed, a flourishing telecom marketplace rife with innovation emerged. We stand at a similar pre-divestiture juncture with Microsoft, in which competition in the operating system market and other ancillary applications markets is artificially limited. A large, robust marketplace of modular, open-code applications stands ready to emerge, but has not. There is hope on the horizon, as Red Hat and other vendors begin to offer integrated packages and components, and as new desktop environments such as KDE and GNOME gain in popularity. Even more striking, IBM is now planning to offer round-the-clock tech support and service for Red Hat Linux. All these steps will help "grow" the market for GNU/Linux and open code software. But we have a long way to go before such "end-to-end" ease of use will be achieved.

F. Can Open Code Software Attract Support from Non-Technical Constituencies?

Ralph Nader became an early advocate of open code software because it radically empowers consumers in many ways. The benefits are not undifferentiated benefits that flow to the aggregate consumer, but highly individualized benefits that may help particular niches of consumers: education, nonprofits, voluntary civic associations, charitable endeavors, professional associations, and business users as well. As open code operating systems become more pervasive, it becomes more plausible to develop more stable, enduring open code platforms and compatible applications. The question is, can non-technical constituencies be made aware of their enormous stake in the emergence of open code software and help spur a demand-pull for it?

Since much of the actual, long-term value of open code software is not immediately understood by the layperson, this could require a lot of missionary work. Indeed, in the short term, open code software is likely to be far too technical and complex for the average user to find comfortable. The systems must evolve more. But this should not deter new initiatives to educate non-technical constituencies about how open code software represents a community- and knowledge-building infrastructure without precedent. It represents a sovereign "media space" independent of proprietary manipulations, arguably one of the most powerful potential forms of user-empowerments in electronic media.

III. STRATEGIES FOR SUPPORTING OPEN CODE SOFTWARE: THE H20 AGENDA

As the preceding analysis suggests, the new software movement is poised to take off and reshape the terms of competition in the software marketplace (and much else besides). But to do so, it must overcome a variety of challenges from proprietary competitors and develop its own capacities as an unconventional software development and distribution system. No one can or should dictate to this community what it should do; its indispensable strength is its participants' sense of ownership and support. Yet new leadership from an independent, movement-owned organization could help consolidate diffused energies, accelerate the adoption of existing open-source software, pro-actively develop new strategic directions, and reach out to a broader array of computer users, legislators and the public.

H20 proposes to play such a role through a number of interrelated initiatives. The Project will provide a useful focal point for research, discussion, policy analysis and strategic planning for the new software movement. It will be more feasible to air and debate disparate perspectives within this community and develop new syntheses, consensus and action plans.

H20 will reach out to non-technical constituencies to inform them of their stake in the development of open code software and develop their own projects based around it. Initially, the Project's focus will be on educational applications of open code software, but we also envision outreach to nonprofit organizations, civic groups, voluntary associations, professional societies, government agencies, and small businesses, among other groups.

Besides some prominent leaders of open-source projects, H20 will have the backing of the Harvard Law School's Berkman Center on Internet & Society, whose six faculty members[23] , eighteen fellows and eighteen affiliated individuals represent a diverse network of professors, students, entrepreneurs, lawyers and "virtual architects." The Center's faculty members regularly teach courses of major issues facing the Internet. It has hosted such major conferences as the Internet & Society Conference in May, 1998 and the November 1998 conference of ICANN, the Internet Corporation for Assigned Names and Numbers, which is in the process of becoming a new governing entity for the Internet.

A. ARTICULATING THE PUBLIC'S STAKE IN SOFTWARE AND THE INTERNET

As open code software gains in popularity, it will face any number of significant challenges that will require sophisticated analysis based on timely empirical knowledge. This section describes some of the key intellectual projects that H20 will facilitate to help expand the new software movement.

1. Develop richer theoretical and strategic analyses to explain the power of open code software development.

One of the most important challenges facing the open code software movement is to refine and deepen its strategic intelligence. It is paradoxical and disturbing, for example, that two of the most thorough, timely and hard-nosed analyses of the open code software movement - the Halloween documents - were generated by Microsoft. To be sure, there are a fair number of academics and computing iconoclasts who are highly informed, capable analysts. But this work is diffused and needs greater sustaining support.

The work of open code opinion-leaders would be enhanced if they had more opportunities to meet, converse informally, plan strategically, develop personal relationships, cultivate new institutional collaborations, organize to address common goals, and communicate them with larger constituencies and the general public. The kinds of interpretive syntheses prepared by the Raymonds, Stallmans, Torvalds, Lessigs and Barlows need to be supported, showcased and brought to the attention of mainstream culture. H20 aspires to offer such support to the new software movement.

Raymond, Stallman, Valloppillil and others have offered important conceptual critiques of the open code software process, but these are really opening skirmishes in a much larger, ongoing intellectual inquiry. There is a need to probe the strengths and limitations of open code software development, and to devise new theoretical tools for understanding it. Received theories of intellectual property and economics, for example, simply cannot explain the "gift cultures" represented by open code communities. Nor can they explain why the weak intellectual property protection of the HTML protocols for the World Wide Web utterly demolished the closed, proprietary approaches by AOL and CompuServe, producing one of the most phenomenal publishing successes in history.

The proprietary world clings to archaic economic theories and legal protections despite the onslaught of a new computing paradigm, even as the new paradigm, awakening to its actual powers, struggles in fits and starts to understand itself. For example, information industries objected so strenuously when government agencies began placing so much taxpayer-generated information on websites, for free. Yet the dire predictions proved false: information vendors have migrated into new lines of value-added information synthesis that complement the government's free disclosure of information. The government stimulated the creation of new niche markets and innovation while shutting down a subsidized, socially wasteful market whose functions could be better and more cheaply served by government itself.

We need more analysis of market transformations that have public implications so that the public and policymakers can make more informed, independent assessments. This would include new economic theories about information technologies and products; new antitrust theories that take account of the novel tactics that can stymie competition; and new analyses of how interoperability in hardware is becoming more critical as software functionality migrates to proprietary hardware interfaces.

To be sure, information theorists are developing new approaches, exemplified by Shapiro and Varian's Information Rules: A Strategic Guide to the Network Economy.[24] But the new theorists do not necessarily regard consumers, democracy, education or the public interest as their foremost concerns. Nor do many of the emerging theories seek to expand public interest values into new realms. Why, for example, shouldn't the principles of open standards be applied to the desktop computing environment, with the commoditizing of the operating system? The open code software movement needs this kind of serious strategic intelligence capacity.

2. Explain the democratic, civic and consumer implications of open code software.

Although the new software movement is gaining visibility, there is relatively little understanding of how open code software empowers consumers and citizens and why it is important to our democratic culture. New analysis and public education are needed to explain the pro-consumer benefits of open code software, both in comparison to existing proprietary products (quality, reliability, flexibility, etc.) as well as in neutralizing the anti-competitive behavior of proprietary software makers, especially Microsoft. This analysis needs to be developed and popularized so that the latent constituencies with a stake in open code software - nonprofits, government, education, etc. - can begin to understand their long-term interests and act accordingly.
The public sector in particular needs to understand how its considerable purchasing power could help "grow" this market and expand its benefits; how open code software could provide the scaffolding for a new "media space" for the civic and education sectors; and how software design is a legitimate realm of public concern.

3. Develop a new taxonomy of intellectual property and software development models.

Although the General Public License (GPL) and Berkeley Software Distribution (BSD) license are popular licensing models, there are a wide variety of open code software licenses in use, many of which blend arcane proprietary goals with open-source development freedoms. It would be useful to develop a detailed taxonomy of this confusing legal domain so that the major patterns of development and their legal and market implications could be better understood. For example, which licenses are compatible with each other? Could a browser that supported commercial technologies (like Java, RealAudio and any number of plug-ins) be protected under GPL? Will the implementation of Harmony as an alternative to Qt succeed, and will the non-GPL open-source licensing model for Qt succeed ?

On a more ambitious level, it would be helpful to fortify the concepts of copyleft with stronger standards under federal copyright law. Why shouldn't the law validate and recognize the GPL and perhaps offer extra civil and criminal sanctions for those who seek to deform its provisions? As an artifact of another era, copyright does not recognize alternative incentives for artistic and creative expression, the social recognition and prestige that occurs in gift-culture communities. It would be useful to develop new legal doctrines that might adequately recognize and protect these motives so that they might have a modicum of legal standing.

4. Examine the threats to open standards and interoperability in the Internet and other computing systems.

The fundamental strength of the Internet as a communications tool and marketplace is its open standards and protocols. Everyone can theoretically compete and communicate without artificial gatekeepers. The various bodies that have given rise to the Internet and assisted with its governance - ARPANET, IETF, IANA (now ICANN) -- have all honored openness and consensus. It is important that this tradition continue and that no "public space" or node of the Internet be controlled by a single vendor or oligopoly of vendors. Yet the threat of a balkanization of the Internet's open standards is growing, as exemplified by the receding government role in supervising the Internet and in the laissez-faire policies proposed by the Department of Commerce and former White House advisor Ira Magaziner.

Ensuring open standards and interoperability is exceedingly difficult when so many of the issues are highly technical and the industry/Internet decisionmaking bodies are not obliged to follow the standards of openness, citizen access and due process that a government body must observe.[25] Even if open access were legally required, the user community cannot readily afford to participate on an equal footing with industry representatives, with all the travel, sustained research, writing and networking that meaningful participation entails. This lamentable situation means that software companies can sometimes "lock up" software protocols that rightfully should be subject to open discussion, review and perhaps modification.[26]

H20 and its network of friends would seek to fill this void by participating in standards deliberations, preparing important technical analyses from the public interest perspective, and alerting the larger public to their stake in the process. It would be a key priority of H20 to monitor the status of open standards and interoperability, identify the primary threats to those goals, and propose steps to ensure that no one captures the key public spaces or interfaces of the Internet.

5. Identify barriers to market entry for free software operating systems.

Under its consent order with the Justice Department, Microsoft cannot prohibit OEMs (original equipment manufacturers) from selling other operating systems as pre-installed software. But that agreement only covers Windows 95, not Windows NT, and applies only in the United States. There are allegations that Microsoft's contract with an OEM in Europe prohibits it from advertising or marketing other operating systems, such as GNU/Linux.[27] The Halloween II document suggested that Microsoft could reduce the price of its NT software, if needed, to pressure OEMs not to pre-install GNU/Linux.

This is only the most egregious known example of how the proprietary world might seek to impede the growth of open code software. It would be useful to explore what other discriminatory barriers to entry may exist and bring them to the attention of Justice Department officials and the computing public. Neither Microsoft nor other companies should be allowed to use their market power to squelch the marketing or use of open code software alternatives.

6. Explore alternative methods of financing the development of free software.

The free market/libertarian ideology that prevails in high-tech circles reflexively dismisses the idea of government playing a salutary role in the marketplace. But, of course, government support was indispensable for the development of the Internet (via ARPANET) and the Netscape browser (derived from the government-funded MOSAIC software). It may not be coincidental that as public sector involvement in the Internet and software declines, the incidence of anti-social, anti-consumer activities (e.g., industry consolidation leading to less competition and choice; Microsoft's anticompetitive tactics; bolder attempts to privatize open standards, etc.) seem to be increasing.

A strong case can be made that government-mediated support for software and networking innovations has an invigorating effect on the larger marketplace. There is no reason why government should not have a hand in actively promoting open code software as an augmentation/pacesetter to the commercial sector's products. The impact on market competition, innovation and consumer well-being is simply too great.

Many feasible schemes are imaginable. James Love of the Consumer Project on Technology, the Nader-affiliated group, has proposed a 1-2% vendor tax on commercial software that would be used to finance the development of GPL-licensed open code software at state universities. Internet innovator Carl Malamud has proposed a tax credit scheme for the same ends. Students could receive work-study support; the software marketplace could be infused with fresh ideas and new programming talent; and the public would get free software and new tools for combating anticompetitive practices in the marketplace. Other financing schemes could include funding by consortia of industry, universities or Internet service providers, which might find it useful to support infrastructure costs to help keep GNU/Linux and other open code software competitive. All these parties, after all, benefit from the open code software that currently makes the Internet function so well.

7. Advocate greater public access to government information.

It is a truism that information is the currency of democracy. Yet even as the Internet is becoming a common tool in the conduct of everyday affairs, many parts of the federal government - and state governments as well - continue to resist putting useful information resources online. The failure to do so is sometimes an attempt to stifle public scrutiny and participation; other times information is simply privatized, unnecessarily enriching private vendors at the expense of taxpayers while diminishing democratic accountability.

Several areas deserve attention. The NTIS, National Technical Information Service, has become a multimillion subsidy to information vendors at the expense of ordinary citizens who cannot afford to buy NTIS documents or data tapes created by their own tax dollars. Releasing this information would not only be the fair thing to do, it would invigorate scientific and technical debate on issues that are otherwise dominated by well-heeled businesses and institutions. Court and administrative agency transcripts are another body of government information that ought to be put online. Currently, however, this information is owned by the official transcribers, preventing citizens from following court and federal agency proceedings that they are financing, many of which are of great public interest.

Many congressional records, too, such as committee reports and legislative mark-ups, are not available on the Internet, notwithstanding the speed with which Congress put the Starr report on the Internet. Finally, Freedom of Information Act requests that do not involve privacy concerns could and should be placed on the Internet. Not only would this make vast amounts of the public's information readily available, it would save the government considerable sums by eliminating costs of processing duplicate FOIA requests.

B. CREATING A SOFTWARE USERS' ALLIANCE

It is important that open code software developers and users have hospitable forums in which to discuss common concerns. They also ought to have the capacity to organize themselves to assert their strategic, legal and political interests as they are affected in the marketplace, industry decisions, public policy and elsewhere. Section B outlines several strategies by which H20 could foster these goals.

1. Convene programmers to help develop new open code programs for niche constituencies.

Open code software programmers do not necessarily need any guidance from H20. On the other hand, people who create open source code software do not necessarily consider novel, transforming applications of their talents. The leadership of a Torvalds or Stallman or Allman is needed to propose a vision and nourish the implementation. H20 could play a valuable role in convening developers to contemplate larger, more ambitious applications of their expertise for students, educators, people of ethnic heritage or narrow affinity groups.

H20 envisions a role of catalyzing educational, business, governmental, and cultural institutions to explore how open code software can help them. Small investments by a number of individual leading institutions can help develop a common, cost-efficient software platform that serves the needs of the Internet community at large.

2. Revamp government and education purchasing to support open code software.

It is not widely appreciated how government, using its own immense market power as a consumer, can dramatically affect the levels of innovation, quality and price in the marketplace. Federal, state and local governments spend more than 18% of the nation's GNP, representing a powerful, largely untapped force for shaping the quality of products delivered. This can be seen in how federal procurement finally persuaded automakers to bring air bags to market. It can also be seen in government-led expansion of markets for recyclable products, chlorine-free paper, longer-lasting road pavement and countless other innovations. Nader, a champion of the strategic uses of government purchasing power, explains: "If properly deployed, government's massive pool of organized purchasing power can jump-start the adoption of new or dormant products and technologies to promote higher productivity, greater human safety and a cleaner environment."[28]

Government agencies typically buy off-the-shelf software products instead of issuing detailed performance specifications to would-be vendors. Some agencies actually specify the vendor - i.e., Microsoft - rather than the generic functions that the software must meet. The federal government's short-term approach to its routine software purchasing means that government wastes lots of money on deficient proprietary software rather than using its imagination to reap better open code software that will not be subject to costly proprietary restrictions and upgrades. Governments in developing countries are starting to realize that the licensing fees of U.S. software vendors are simply too expensive (especially as trade agreements are enforced more vigorously) when there are cheaper, technically superior alternatives such as GNU/Linux. Mexico is reportedly shifting more than 140,000 computers in educational settings to GNU/Linux for this very reason.

Higher education, also, has much too much leverage - student-consumers, programming talent, institutional purchasing dollars - to accept the limited choices offered by proprietary vendors when open code software offers greater long-term cost savings, flexibility and stability. Instead of flexing their muscles in this direction, however, many major universities allow themselves to serve as marketing and recruitment vehicles for Microsoft, accepting discount software deals in return for exclusive access to a future consumer base and programmers' mindshare. But even these seemingly smart deals can prove to be more expensive over the long term as licensing regimes are changed to extract higher prices once the company has a monopoly stranglehold.[29] It is entirely appropriate and consistent with higher education's public mission and taxpayer expenditures that the software it supports be open code software. New initiatives in this area could yield stunning benefits to higher education over time.

For those who object that government has no business short-circuiting the marketplace or the verdict of open code communities, the answer is that government has to buy something. Why shouldn't it be the best-performing open code software, as determined by performance specs and open competition? The case for more intelligent, innovation-minded government purchasing -- rather than the slavish acceptance of brand-names - is compelling. But first, the case must be made in a more detailed way and presented to Vice President Gore, the GSA Administrator and the public.[30] If successful, even in small instances, government purchasing of open code software could have powerful psychological and economic repercussions as smaller purchasers follow suit and expand the open code community/market further.

3. Assert consumer rights in the software & Internet marketplace.

While individual users enjoy remarkable new capabilities through the Internet, their ability to assert their common strategic interests in the new electronic marketplace and public policy issues is limited. Users have few forums or resources that can effectively advocate their collective interests in consumer rights, intellectual property, online privacy, access to government information, interoperability of computer systems, among other issues. H20 plans to be a convenor and supporter of diverse efforts to promote consumer rights in the software/computing/Internet environment.

The need for sophisticated legal advocacy in this area is exemplified by the little-known revisions being prepared for Article 2B of the Uniform Commercial Code, a cornerstone of American business law that governs the sale and lease of software, databases and information. The American Law Institute and the National Conference of Commissioners on Uniform State Laws are redrafting this provision of the law which, if adopted by all 50 states as intended, would become controlling law for transactions constituting 30% or more of the U.S. economy. The redrafting of the law is dominated by industry lobbyists, leading to the not-surprising result that the new draft language will allow sellers to legally bind consumers to surprising, unreasonable legal provisions buried in pages of legalese so long as the user clicks the "I agree" box on a software program or web site. As it happens, this proceeding is being monitored by Harvard law professor Lawrence Lessig[31] and by Todd Paglia of the Consumer Project on Technology.[32]

Smart, sophisticated user representation in obscure, industry-dominated arenas remains fairly rare. Yet as the call for industry self-regulation grows - in standard-setting, international trade, consumer protection, privacy, intellectual property law, and many other realms -- there are serious questions whether governance of the Internet will become privatized and our democratic traditions abandoned. It is vital that user/citizen interests be asserted more forcefully and intelligently so that governance can be truly responsive to sovereign user interests and not just business interests.[33]

4. Develop and "manage" a brand identity for open code software.

As discussed in Section II, the new software movement must begin to develop a brand identity if it hopes to expand its influence in the mainstream computing environment and enlist the support of non-technical constituencies. This need not imply competing with the proprietary world's marketing budgets and advertising reach - a style of battle that the open code movement can never win. Indeed, the open code movement's grassroots identity and gift-culture ethos call for a different style of marketing. Yet marketing, nonetheless, is needed. It is presumptuous and premature to say what such a campaign might consist of. But H20 does hope to convene leading actors in the open code world to facilitate a needed discussions about this topic.

Part of projecting a brand identity is knowing what open code products are available and helping users learn about them and acquire them. To this end, H20 hopes to facilitate the creation of a new clearinghouse for open code enthusiasts. There are hundreds of open-source programming communities for hundreds of different purposes. Assembling a more comprehensive inventory of these projects and making them more readily available would have an immense catalytic influence.

5. Serve as a repository and users' forum to sustain and improve abandoned software programs.

As the pace of innovation in computers and software accelerates - and as corporate mergers and acquisitions reconfigure the marketplace - many vendors are abandoning popular and entirely effective software programs. Popular programs such as Sidekick, Xtree, EchoPro and soon, it is predicted, Eudora, are being thrown on the dustbin, forcing satisfied users to buy new software. H20 proposes becoming a standing repository for these programs by becoming the legal owner of the source code and, where possible, helping interested user groups to organize themselves to sustain legacy products. The vendors donating the software could receive tax deductions for their beneficience, and the software could be put into the public realm through a General Public License. Interested users could keep alive programs that they like, adapt them to new hardware systems and improve them in other ways.

Having H20 serve as a repository for legacy software could also be of immense value to libraries and archives. "Digital files are utterly brittle," writes futurist Stewart Brand. "They're complexly immersed in a temporary collusion of a certain version of a certain application running on a certain version of a certain operating system in a certain generation of a certain box, and kept on a certain passing medium such as a five-inch floppy."[34] Now that the half-life of new computer technology is about three to five years, professional librarians and archivists are facing the practical consequences of relentless technological innovation: millions of documents cannot be easily accessed. By providing a practical means for old software systems to continue to function, H20 would help meet the formidable challenges of document preservation and access.

CONCLUSION

Today, at the dawn of the Internet as a popular medium, as major corporations vie for hegemony over the new communications infrastructure and standards, open code software holds great promise to forge a new, more socially constructive path. It offers the chance to empower consumers, nonprofits, education and various civic segments of American society. It allows for more open, flexible architectural designs for markets, communities and cultures. It enables the construction of cheaper, more durable new media platforms which can accommodate scalability and innovation with great ease.

Capitalizing on this historic opportunity requires the commitment and support of the nation's foundations, one of the few American institutions with the resources, stature and independence to provide risk-taking leadership at this time. This proposal is an attempt to open a dialogue and to make the case for founding a new organization, H20, to facilitate the growth of the open code software movement.

We are mindful that not all moments in history are alike. Lost opportunities are not easily recovered. American life might have evolved along a very different path in the 20th Century if nonprofit advocates had won the debate between 1928 and 1935 over who would control the public airwaves.[35] Commercial broadcasters prevailed, of course, and citizens, educators, labor, religious groups, artists and untold other Americans were largely banished from radio and television. The medium came to be identified with a narrow range of (commercial) formats and untold alternative possibilities were simply never developed.

Unlike the struggle to control broadcasting, the coming struggles over software design, Internet governance and the conduct of electronic commerce will not be resolved through public policy alone. Outcomes will also pivot upon the ability of software users and developers to assert their sovereign interests in the marketplace, industry fora, standards-setting bodies, legal venues, the trade and general press and other vectors of influence. In this early stage of the Internet and software design, as critical decisions are made, it is important that we strive to take risks on the best future we can imagine if only because there may be highly unpleasant, irrevocable costs for not doing so.

David Bollier
March 10, 1999

David Bollier is an independent journalist and consultant based in Amherst, Massachusetts, who writes frequently about the social and democratic implications of the electronic media. A student of citizen advocacy, progressive politics and cultural change, he has worked for the past fifteen years with television writer/producer Norman Lear on assorted non-television projects; with Ralph Nader on a number of civic empowerment/public policy initiatives; and with several foundations and activists who know how to grow new vectors of possibility. In recent years, Bollier has proposed a strategic agenda for reinventing democratic culture using new electronic media (http://www.netaction.org/bollier/index.html); chronicled socially visionary models of business management in his book Aiming Higher; and synthesized the economic, social and political reasons for curbing suburban sprawl and rejuvenating urban regions (http://www.sprawlwatch.org). Bollier can be reached at bollier@essential.org.

NOTES

1.
Much of this debate revolves around what kinds of software licensing and distribution ought to prevail, and what terms of engagement with the proprietary software world should be entertained. Richard Stallman, a leading software development, prefers the term "free software" to stress the freedom to use source code as one sees fit and the freedom to use derivative programs as well, without the customary restrictions that copyright law imposes. Others, such as Eric Raymond, prefer the term "open source software" to stress the accessibility of the source code even if proprietary derivative versions (having closed, copyrighted source code) are allowed. It should be noted that "freeware" and "shareware" are not appropriate synonyms for "free software" because "freeware" does not necessarily have open source code; it can simply be a free or promotional version of a program whose binaries (but not source code) are accessible to users. A review of the many strands of the new software movement can be found in Charis DiBona, Sam Ockman & Mark Stone, editors, Open Sources: Voices from the Open Source Revolution (O"Reilly & Associates, 1999).

2. See "Only the Free World Can Stand Up to Microsoft" at http://www.contex.com/ftwalk/FreeWorld.html, a site that is currently inoperative but seeking a new home. Hull's critique echoes Ralph Nader's seminal essay, "How to Think About the American Economy," The New York Review of Books, September 2, 1971.

3. A few noteworthy examples: Sendmail is the program that routes over 80% of all email on the Internet. Perl is the programming language that allows dynamic features on many web sites. Apache is the most popular web server software in use on the Internet. BIND, the Berkeley Internet Name Daemon, is the de facto DNS server for the Internet. Mozilla is the open code software used in the popular Netscape browser.

4. See, e.g., David Bornstein, The Price of a Dream: The Story of the Grameen Bank (University of Chicago Press, 1997).

5. See http://www.timedollars.org, and Edgar Cahn and Jonathan Rowe, Time Dollars: The Hidden Currency That Enables Americans to Turn Their Hidden Resource - Time - Into Personal Security and Community Renewal (Rodale, 1992).

6. For more on the Free Software Foundation and GNU Project, see http://www.gnu.ai.mit.edu/fsf/fsf.html.

7. Two excellent profiles of Linus Torvalds and the evolution of GNU/Linux are: Glyn Moody, "The Greatest OS that (N)ever Was," Wired, Issue 5.08, August 1997 or http://www.wired.com/wired/5.08/linux.html), and Josh McHugh, "For the Love of Hacking," Forbes, August 10, 1998. Linux International, a nonprofit organization formed to promote GNU/Linux use, can be reached at http://www.li.org.

8. Tim O'Reilly, "The Open Source Revolution," Release 1.0, November 1998.

9. The latest version of Raymond's essay can be found at http://www.tuxedo.org/~esr/writings/cathedral-bazaar.

10. See, e.g., Robert X. Cringely, Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition and Still Can't Get a Date (New York: Harperbusiness, 1996).

11. The number of copies shipped to customers more than tripled from 1997 to 1998, fueled by anti-Microsoft sentiment, strong performance and low pricing, according to a study by International Data Corporation. See story by Stephen Stankland, CNET News, December 16, 1998, http://www.news.com/News/Item/0,4,30027,00.html?st.ne.ni.rel. Also, The Economist, February 20, 1999, p. 63..

12. See http://www.opensource.org/halloween1.html.

13. See http://www.opensource.org/halloween1.html.

14. Steve Lohr, "The Microsoft Trial Reviews Dispute with Sun Over Java," The New York Times, December 11, 1998.

15. This is based on accounts posted on http://www.slashdot.org by John Bryan, among others.

16. See John G. Spooner, PC Week Online, February 22, 1999, at http://www.zdnet.com/pcweek/stories/news/0,4153,1013959,00.html. /li

17. Ibid.

18. The term, as used in this context, was popularized by Eric Raymond in his essay, "Homesteading on the Noosphere," at http://sagan.earthspace.net/~esr/writings/homesteading/.

19. See, e.g., Ben Elgin, "Netscape to Cut Mozilla's Cord?" in Sm@rt Reseller, at http://www.zdnet.com/zdnn/stories/news/0,4586,2176488,00.html.

20. See Deborah Gage, "Sun (Almost) Opens Solaris," in Sm@rt Reseller, at http://www.zdnet.com/zdnn/

21. stories/news/0,4586,2215357,00.html. /li

22. See Robert X. Cringely, "The Pulpit," at http://www.pbs.org/cringely/pulpit/pulpit19980924.html.

23. See http://www.opensource.org/osd.html.

24. Charles Nesson, Lawrence Lessig, Jonathan Zittrain, Arthur Miller, Charles Ogletree and Terry Fisher.

25. Carl Shapiro and Hal R. Varian, Information Rules: A Strategic Guide to the Network Economy (Boston: Harvard Business School Press, 1998).

26. These include the rules set forth by the Administrative Procedures Act, the Federal Advisory Committee Act and the Government in the Sunshine Act.

27. An excellent example of this syndrome occurred in January 1999 when Microsoft was awarded a patent covering the use of style sheets in electronic publishing. With no clear notification to the W3C community or the wider computing world, Microsoft succeeded in essentially privatizing standards that resemble those used in the World Wide Web Consortium's Cascading Style Sheets (CSS) and eXtensible Style Language (XSL) standards. See http://www.camworld.com/misc/mscsspatent.txt or Antone Gonsalves, "Questions Raised About Microsoft Patent," PC Week Online, at http://www.zdnet.com/pcweek/stories/news/0,4153,1013845,00.html /li

28. See Letter to the Department of Justice by the Consumer Project on Technology, June 15, 1998, at http://essential.org/antitrust/ms/jkjun151998.html.

29. Ralph Nader, Eleanor J. Lewis and Eric Weltman, "Shopping for Innovation: The Government as Smart Consumer," The American Prospect, Fall 1992, pp. 71-78. Nader's Government Purchasing Project is a leading agitator for government procurement to promote environmentally sensitive and energy-efficient technologies. See http://www.gpp.org.

30. See Nathan Newman's report, "Microsoft Goes to College" in the October 4, 1998, online issue of NetAction's M$ Monitor, at http://www.netaction.org.

31. See the NetAction report by Mitch Stoltz, "The Case for Government Promotion of Open Source Software," at http://www.netaction.org/opensrc/oss-report.html.

32. See http://www.thestandard.com/articles/display/0,1449,2583,00.html?01.

33. See http://www.cptech.org/ucc.

34. An excellent critique of this issue is made by Lawrence Lessig in "Governance," a keynote address to the CPSR Conference on Internet Governance, October 10, 1998, at http://www.cpsr.org/conferences/annmtg98/.

35. See, e.g., Robert W. McChesney, Telecommunications, Mass Media and Democracy: The Battle for Control of U.S. Broadcasting, 1928-1935 (New York: Oxford University Press, 1993).

36. See, e.g., Robert W. McChesney, Telecommunications, Mass Media and Democracy: The Battle for Control of U.S. Broadcasting, 1928-1935 (New York: Oxford University Press, 1993).

OPENCODE

THE POWER OF OPENNESS

Why Citizens, Education, Government and Business Should Care About the Coming Revolution in Open Source Code Software

A Critique and a Proposal
for the H20 Project

BY DAVID BOLLIER

http://www.opencode.org

Introduction

As software and networking technologies rapidly insinuate themselves into the deepest reaches of American commerce, culture and governance, the architecture of our democratic society is being transformed. One lesson that is becoming clearer is that the design of hardware and software and the governance of the Internet matters. These issues can profoundly affect competition and innovation in markets, the ability of universities, libraries and nonprofits to pursue their missions, and the control that individuals can exercise over their lives.

Within the past year, a number of forces have converged to suggest the socially constructive potential of software whose design code can be freely accessed and modified by computer users. It is a complex story that is still unfolding and known chiefly in computing/Internet circles. As we will explain in Section I, a growing grassroots movement on a global scale is challenging proprietary models of software development by generating superior, more reliable software that is far cheaper and even free.

The implications are not just technical but economic, political and cultural. A new software movement based chiefly on free, open access to the source code of software, is showing its tremendous power to fortify user sovereignty in the computing/Internet marketplace. This movement represents one of the most novel, potentially powerful expressions of the consumer movement in a generation. Over time, if fully developed, the new models of software development could produce innovative, cost-efficient software at much less cost than today, a capacity that could particularly benefit the voluntary, academic, nonprofit and professional communities. It could also help check the excesses of a market-dominated culture by fortifying these "gift culture" communities while mitigating worrisome concentrations of corporate power in the software industry.

But the rich latencies of this Internet-facilitated phenomenon may never develop if a new kind of networking leadership does not coalesce to assert the important values that can only flourish in an environment of openness. The user community and many non-technical constituencies must begin to identify and advance their strategic interests in open code software. Such a mobilization of resources is especially needed because many segments of the software and computer industries seem committed to containing the expansion of open code software, but, for now, have not consolidated enough to develop a unified response.

In the meantime, fissures among the proponents of open source code software may complicate the forward momentum of this movement. One branch, the Free Software movement, sees distinct moral, social and civic value in the source code of software being legally available to anyone in perpetuity. Another branch, the Open Source movement, is more concerned with the technical superiority of open source code software and its intriguing commercial possibilities. Rather than privilege one or the other branch of the movement - whose strategic objectives both overlap and diverge -- this essay will refer to both collectively as "open code" (not "open source") unless one or the other is specifically intended. In the interest of inclusiveness, this text will also use the term "new software movement" to refer to new non-proprietary models of software development and alternative intellectual property regimes.[1]

To the layperson, it may not be immediately apparent why the new software movement holds rich promise. This proposal seeks to explain why its surprising emergence into the mainstream in 1998 is so significant and how, with timely and strategic assistance, open code software could evolve into a powerful new platform for the reinvigoration of the non-commercial civic sector in American society. By radically empowering computer users, it could help rejuvenate our democratic culture, improve education, catalyze a more consumer-responsive economy and ensure fairer, open governance of the Internet.

To reap the full potential of the open code revolution, we propose creating H20, an independent nonprofit organization to help foster the development and usage of the new software. We use the metaphor of water, H20, to emphasize that software is increasingly indispensable to life, at least an enfranchised civic, political, economic and social life. If software is to truly improve these domains, it must be capable of circulating freely so that it can nourish the fundamental values of an open society: democratic participation, social equity, equal opportunity and educational achievement, among others. Openness is a virtue in software development not just because it tends to produce superior product, but equally because it fortifies free market competition and democratic principles.

This essay explains why open code software is so important, especially for various non-technical constituencies, and how a new organization - H20 - could help promote new development and usage of open source code software. This document is deliberately aimed at the layperson as much as the computer sophisticate because a new and broader conversation must be started, one that considers the far-reaching implications of open code software for how we shall govern ourselves, improve education, foster innovation and economic growth, and protect the sovereign interests of citizens and consumers. This text, then, is intended for anyone interested in these realms as well as for influential leaders of the foundation community who could catalyze some powerful changes by fostering the development of open source code software.

The plan of this essay is as follows:

I. THE RICH PROMISE OF OPEN CODE SOFTWARE

The "deep architecture" of global society is increasingly predicated on software, computers and telecommunications. In diverse permutations, these technologies are changing the skeleton, sinews and central nervous system of economy, democracy, organizations and public life. The kinds of knowledge that can be created and shared, the ways that markets are structured, the functioning of democracy, the kinds of social connections and identities that we develop, the kinds of public cultural spaces and conversations that can flourish: all are affected by the ongoing revolution in computers and electronic communications, especially the Internet.

Since software lies at the heart of so much of our new societal architecture, it is important to understand what forces are shaping the structure and evolution of software design. Software resembles a genetic force; entire worlds are enfolded within its embryonic design. As a theoretical matter, software is highly malleable, even protean. In practice, commercial software companies generally manage the design and evolution of their products to enhance their competitive advantage, a calculus that may or may not serve users' best interests. Online commentator Tom Hull explains:

> [I]n the world we live in, production is highly organized and efficient and commands enormous financial resources and seductive powers of persuasion, while demand is fragmented, uninformed and powerless. While consumers can still kill a product that they have no desire for, they are nearly powerless to direct or even influence the detailed designs of those products. For software products, consumers can only choose among a given set of alternatives, which are extremely complex, dauntingly impenetrable, and generally designed more for the company's anticompetitive purposes than for the user's tasks. [2] [I]n the world we live in, production is highly organized and efficient and commands enormous financial resources and seductive powers of persuasion, while demand is fragmented, uninformed and powerless. While consumers can still kill a product that they have no desire for, they are nearly powerless to direct or even influence the detailed designs of those products. For software products, consumers can only choose among a given set of alternatives, which are extremely complex, dauntingly impenetrable, and generally designed more for the company's anticompetitive purposes than for the user's tasks. [2]

The revelations of the current antitrust trial against Microsoft dramatically illustrate this point. The U.S. Department of Justice alleges that Microsoft has used its dominance of the computer operating system market to unfairly restrict consumer choice of other operating systems, browsers and other software applications. As the maker of more than 90% of all desktop computer operating systems, Microsoft has been able to charge higher prices for its software through restrictive licensing agreements with computer equipment makers, despite the availability of superior alternatives. Through a strategy described at the Microsoft antitrust trial as "embrace, extend and extinguish," the company has taken open standard protocols such as HTML (for web pages), Java (the cross-platform software), RealAudio (the Internet audio software) and QuickTime (multimedia software), and sought to sabotage them by introducing its own proprietary modifications as the de facto standard. Microsoft has also aggressively used a FUD strategy - "fear, uncertainty, doubt" - to scare would-be competitors from entering markets Microsoft wants to dominate. Such aggressive actions have chilled innovation in the software industry by forcing other developers to retreat to market niches where fair competition is possible.

While Microsoft may be a singular force in computing, its alleged tactics are simply an extreme case of what other software companies would likely use as well, if they could. Many software companies aspire to use ingenious proprietary design standards to create or dominate new market niches, "lock in" their technology, and preemptively limit competition and user choices. For the most part, the "seller side" is committed to determining what types of software innovations will be allowed to compete in a given market; what types of licensing arrangements shall govern users; what prices consumers will pay for products whose incremental costs of production are virtually nil; and what websites will be given preferential access or development by the primary Internet gatekeepers/portals.

It is important to understand that software design can be used strategically to shape and control markets and even national cultures. This is illustrated by PICS, the Platform for Internet Content Selection, a sophisticated HTML standard that allows private agencies, governments and corporations to rate and block online content.

Although "content-neutral" in design (it can filter out birdwatching literature as easily as Nazi Party diatribes), the PICS technology is an unparalleled tool for censorship on a global scale. The development of PICS by a small cadre of little-known technicians and business executives suggests how software can be used to "design" the character of markets, political life, and culture. The so-called Y2K or "Year 2000" problem also illustrates how even a small but structurally entrenched problem in software design (the truncated field that bedevils storage and retrieval of the year value for years that do not begin with 19xx) can affect everything from air traffic control and international currency trading to emergency services.

A. The Importance of Source Code in Software

Proprietary source code, protected by copyright law, plays a central role in enabling the "architectural design power" of software. Originally, copyright arose to protect and reward creative expression in printed media in order to assure its public dissemination. But copyright is often misused as a legal tool for withholding creative expression and controlling the terms of competition in a given market. The seller can prevent users from altering the functionality of a software product, for example, or limit its interoperability with other software and hardware. The seller can prevent users from customizing the product to suit their own needs. The seller can ignore known bugs and other product flaws, and in other ways force users to accept unwanted design features in a product.

Historically, software makers have sought to preserve these prerogatives (and thus their relative advantage over competitors and consumers) by vigorously protecting their "source code," the internal blueprint of a software product that is the basis for the "binary code," or binaries, the user-accessible manifestation of a software product that a computer executes. As long as the proprietary source code can be held as a trade secret protected through copyright law, sellers retain greater control than users over the terms of software design, quality control, pace of innovation and use.

However important proprietary control of source code may be in some respects, that control diminishes consumer sovereignty, accountability and choice. In many instances, it results in anticompetitive mischief. This is especially pernicious because, in rapidly changing markets with technically complex products, neither consumers nor antitrust regulators are truly equipped to challenge such marketplace abuses except in the most egregious instances (such as the Justice Department's action against Microsoft), and the ultimate remedies secured may be deficient in any case.

The new software movement offers a radically new basis of competition, innovation and consumer power in the marketplace. Propelled as if from nowhere by converging forces, the new software movement has been catapulted into the mainstream over the past year. This user-driven model being generated by a vast global community of computing irregulars not only rivals proprietary software in quality, reliability, flexibility and price; it effectively "belongs" to the user community and therefore offers a cheap, versatile and durable scaffolding for the self-development of all sorts of user communities.

Just as the cooperative movement has enabled workers and consumers to assert greater control over their economic and personal lives, so the new software movement opens up new frontiers of self-determination. It allows users to pole-vault over the cost inefficiencies, barriers to innovation, consumer manipulation and design rigidities that characterize many proprietary software markets. (Open code communities are able to bypass most of the traditional impediments to coops because the Internet allows incremental and ad hoc participation, participation without regard to geography, avoidance of overhead and management costs, and rapid feedback loops with consumers.)

For all its promise and remarkable resourcefulness, the new software movement faces many serious challenges if it is going to develop a more substantial influence in the marketplace, in life, in cyberspace. Before examining that issue in Section II below, it is important to understand some of the deep dynamics of the new software movement, what makes its software so robust and innovative, and how it could, with the kinds of assistance H20 proposes, enable myriad social, educational and civic benefits.

B. The Growth of the New Software Movement

In the early days of computing, a great deal of software development was a collaborative process carried out by academics and students. Software was seen as the shared product of a community, to which everyone freely contributed and benefited. In this respect, open code software shares the strength and resiliency of the scientific method and Jeffersonian democracy. All procedures and outcomes are subject to the scrutiny of all. Openness allows error to be more rapidly identified and corrected. Innovation and improvement can be more readily embraced. Openness builds accountability into the process of change.

The commercialization of computing in the 1970s and 1980s introduced a new dynamic to software development: a closed, proprietary process that mobilized expertise to develop innovations, bring them to the market, and reward private investors. Bill Gates' entrepreneurial passion was so great he was nearly expelled from Harvard for using publicly funded labs to create commercial software - a violation of the hacker ethic of sharing (and not privatizing) community knowledge. After Gates was required to put his code in the public domain, as free software, he quit Harvard and went on to found Microsoft. The high-tech entrepreneurialism that Gates exemplifies is now the stuff of American legend.

Yet lurking in the shadow of this mighty new industry over which Bill Gates reigns, the free software movement has quietly persisted and grown. Empowered by the Internet, the community of computer aficionados willing to develop, improve and freely share software among themselves has mushroomed. Even as Silicon Valley gives birth to dozens of startup ventures each year, driven in no small part by the connectivity of the Internet, so the hacker community has grown in sophistication, breadth and international scope. It has generated hundreds of top-quality software programs, many of which have become critical operating operating components of the Internet.[3]

What most distinguishes this free software from the off-the-shelf proprietary products is the openness of its source code - and the user's freedom to use and distribute the software in whatever ways desired. Anyone with the expertise can "look under the hood" of the software and modify the engine, change the carburetor or install turbo-chargers. Inelegant designs can be changed, noisome bugs can be fixed (or introduced). Sellers cannot coerce users into buying "bloatware" (overblown, inefficient software packages with gratuitous features), Windows-compatible applications or gratuitous software upgrades contrived through planned obsolescence. Nor are constant upgrades in computer hardware (such as the latest, high-speed Intel chips) required.

In short, open code software allows users to assert much greater control over their computing environment. This is not a matter of customizing a spell-checker; users with access to source code can change basic functionalities of their software and reap significant new efficiencies, often for free. By enabling skilled users to customize software to suit their special needs, open code software represents an entirely new kind of media empowerment. It potentially allows academic specialties, libraries, distance educators, civic organizations, business enterprises and others to develop their own innovative vehicles for sharing and elaborating a common body of knowledge. They do not have to adapt to the design structures, constraints on innovation and licensing/price schemes imposed by proprietary vendors. In this sense, open code software is not just a product, but a new kind of knowledge- and community-building infrastructure.

The benefits that users reap from open code software - customization, innovation, education, security, efficiency, reliability, cost savings - are actually "symptoms" of their collective empowerment as users. By banding together to assert their common interests, open code software users acquire an entirely new dimension of power. They are not merely consumers picking and choosing from the products sellers may choose (or decline) to offer; they are co-producers in the creation of specific software to serve their distinctive needs. The power wielded by the open-code user networks is vastly greater than that which propelled public interest initiatives in other media -- universal service in telephony, weak rights of citizen access in television and radio, a chronically beleaguered public broadcasting system; and local access channels for cable television. All these initiatives have required political interventions, court litigation or significant government spending, which means that have been constantly vulnerable. By contrast, the open code software movement is a sovereign, expansive "political" force in its own right.

C. The Alternative Economics of Open Code Software Development

Although critically aided by the Internet, the power of the new software movement stems from the "gift culture" that lies at the heart of the open code development model. The Internet is a quintessential example of a gift culture. People are willing to make all sorts of useful information available for free, in defiance of orthodox economic "rules" that claim such voluntary behavior can occur only with financial incentives. The Internet is so robust precisely because people are giving of themselves without demanding a specific contractual payback. This is the very essence of community and civility. People are willing to enter into gift economies because they trust that they will at some point share in the "wealth" that the community freely passes among itself - much as an academic community freely shares its knowledge among its members and disdains those who seek to financially profit from the community's shared body of knowledge. The key point is not that the information or software is economically free - or "free beer," as the jibe goes - but that it is freely accessible and amenable to modification by anyone interested in doing so. (There are, of course, other reasons why people contribute software and open code to the public over the Internet. Software companies may want free publicity or a greater influence over user "mindshare"; a software developer may want to showcase his programming prowess; an academic community may wish to collectively share its resources.)

The phenomenon of leveraging the moral commitments of a community of people to achieve economic development and collective improvement is not exotic; it's simply denigrated by mainstream economic theory. Micro-lenders such as Grameen Bank in Bangladesh and South Shore Bank in Chicago have proven that they can make safe, profitable loans to inner-city entrepreneurs and collateral-poor mom-and-pop ventures that conventional banks find risky.[4] How? By realizing how trust, moral commitment to one's peers and a concern for reputation are not only economically valuable but often more powerful and motivational than economic incentives alone. Similarly, the "Time Dollars" program has also shown how moral and social values can be harnessed to achieve "economic" goals. People can earn "time dollars" (one hour, one "time dollar") to babysit, rake leaves or provide legal assistance for other members of the community; the credits can then be used to "buy" services from other participants in the network. With no exchange of money, a considerable amount of economically valuable activity is performed - work that conventional markets are often incapable of eliciting.

So it is with open code software, as facilitated by the Internet. Many of the crucibles that incubate open code software are "gift economies," whose economic development is tightly linked with community-building. By the lights of mainstream economics, the moral and community dynamics that help generate open code software simply do not make sense. But in real life, forget the theories: open code software development works. But however formidable its creative capabilities, open code software will be developed only in those domains where its value is recognized. And many cash-strapped enterprises within academia, the nonprofit world and community organizations - as well as many businesses -- do not yet recognize the remarkable tools they could develop. The leadership and education that H20 proposes could make a significant difference.

Richard Stallman, a programmer at the MIT Artificial Intelligence Laboratory in the 1970s, was one of the first to recognize the tensile strength of what he called "free software," in which "free" refers to the freedom to change the source code, not free in price.[5] For both personal and philosophical reasons, Stallman has strenuously objected to proprietary software because its copyright licensing restrictions limit his personal freedom to share programs with his friends and co-workers, to change the programs as he wishes, and to distribute improved versions that will benefit the larger community. He considers copyright control of software to be philosophically repugnant because it cripples the very advantages of digital technology -- its flexibility and ease of reproduction and sharing -- and encourages personally intrusive kinds of enforcement. As his peers trooped off to make fortunes in the emerging computer software industry in the early 1980s, Stallman instead founded the Free Software Foundation. The visionary project (which earned Stallman a MacArthur "genius" award grant) is dedicated to developing and promoting free software.[6]

D. The GNU Project and the GPL

Two innovations developed by Stallman and his colleagues have special relevance to the new software movement. First, Stallman in 1983 launched the GNU Project, a collaborative endeavor to develop a free Unix-like operating system (free in price as well as open source code). ("GNU" is an acronym for "GNU's Not Unix," a recursive pun; Unix, of course, is the open standards-based operating system developed by Bell Labs in the 1970s. Unix was the original operating system of the Internet and of governments and academic institutions, a role that it continues to play to this day.) A primary motive behind the ambitious GNU project, Stallman explained, was to "bring back the cooperative spirit that prevailed in the computing community in earlier days -- to make cooperation possible again by removing the obstacles to cooperation imposed by owners of proprietary software." Ultimately, Stallman wants to supplant proprietary operating systems by making them free, open-source code commodities. By the 1990s, the GNU Project had found or written all the major components of an operating system except one, the kernel.

Fortuitously, a free kernel developed by a Finnish student, Linus Torvalds, and greatly improved by a large self-organized network of volunteers, appeared on the Internet in 1991.[7] This kernel, when combined with the Unix utilities developed or collected by the Free Software Foundation over the past twenty years, "was like setting a match to dry leaves, creating an entirely new and completely open operating system overnight," according to one account. The conjoining of the GNU system with the so-called Linux kernel (Linus + Unix = Linux) has largely fulfilled Stallman's vision of a well-designed, free Unix-like operating system that can run on diverse hardware platforms. There are other free operating systems that command great respect among software developers, yet none has achieved the huge popularity that GNU/Linux has achieved in recent years. It has been a "stealth" operating system because technophiles in major companies often secretly used the software, knowing that their bosses would disapprove of the use of a "free" operating system of unknown provenance in mission-critical applications. Now that the reliability, versatility and price of GNU/Linux has received mainstream validation, a juggernaut has been unleashed. It has become the fastest-growing flavor of Unix and a preferred operating system in such diverse organizations as Boeing, Northern Telecom and NASA. It is also noteworthy that many enterprises and universities in Mexico, China, France, Australia and eastern Europe - fed up with paying expensive licensing fees for less reliable, versatile proprietary products -- are adopting GNU/Linux.

GNU/Linux usage received its biggest boost in 1998 when the press discovered Linus Torvalds, making him the iconic underdog of the computer world. Meanwhile, over the past year, one major software developer after another - IBM, Oracle, Corel, Inprise, Informix, others -- has announced they will port software applications to run on GNU/Linux. A number of vendors -- Red Hat, Caldera, SuSE, Pacific Hi-Tech - have begun to sell GNU/Linux along with documentation and technical support, attracting significant investments from the likes of Intel and Oracle. It is estimated that more than seven million people now use GNU/Linux as their operating system.

GNU/Linux might never have emerged but for Stallman's second innovation: the GNU General Public License (GPL), sometimes known as "copyleft." Stallman astutely realized that simply putting free software into the public domain was not enough, because anyone could make minor changes in a program and then copyright it, converting it back into a proprietary product. Without some legal vehicle, the benefits of free software could be privatized and withheld from the community of users. So Stallman developed the GPL, which is essentially copyright protection with special contractual terms. "To copyleft a program," writes Stallman, "first we copyright it; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify and redistribute the program's code or any program derived from it, but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable. Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom."

The GPL offers the greatest legal assurance that open source code will remain free and available to all, and that no company will appropriate the property for itself. But there are other copyright licensing variations. The BSD-style licenses -- based on the Berkeley Standard Distribution of Unix -- gives software users the option of creating derivative versions that can be copyrighted and made proprietary -- without revealing the source code. When Netscape released the source code of its Communicator browser in early 1998, it issued a Mozilla Public License, or MPL, which steers between the terms of the GPL and BSD licenses (private derivative versions can be made but any changes to source code covered by the MPL must be made freely available on the Internet).
Licenses that allow the privatization of the public pool of source code - thus creating a new proprietary line of development, often at the expense of the public pool -- is called "code forking." Stallman and other programmers are adamant about the principles of the GPL because code-forking vitiates the momentum of free software development. "Some in the open source community resent third parties taking from the public pool of software without contributing," writes publisher Tim O'Reilly in a major review of the open source movement. "In economies, this is called the free-rider problem. But despite the lack of a mandate, voluntary cooperation abounds."[8]

This philosophical divide is sometimes a contentious one, aggravated by the growing interest of technology companies in open code development models. It remains unclear if the powers of an open code "gift community" can be harnessed by software companies that answer ultimately to investors, not that community.

E. The Cathedral and the Bazaar

Although Stallman's GNU Project and Linux are two sentinel developments in the new software movement, they by no means define it. They are simply two of the largest manifestations of a more significant development, the open, Internet-facilitated process of software development. The idea that thousands of computer volunteers working together through the Internet might actually produce a sophisticated operating system that could out-perform proprietary software seemed ludicrous until GNU/Linux began to emerge. Open source theorist Eric Raymond explained the anomaly and thereby gave it wider credibility through a seminal online essay, "The Cathedral and the Bazaar," which first appeared in April 1997.[9] Raymond writes:

> Linux overturned much of what I thought I knew. I had been preaching the Unix gospel of small tools, rapid prototyping and evolutionary programming for years. But I also believed there was a certain critical complexity above which a more centralized, a priori approach was required. I believed that the most important software (operating systems and really large tools like Emacs) needed to be built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time.
>
> Linus Torvald's style of development -- release early and often, delegate everything you can, be open to the point of promiscuity -- came as a surprise. No quiet, reverent cathedral-building here -- rather the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from anyone) out of which a coherent and stable system could seemingly emerge only by a succession of miracles.

Raymond's great contribution was to explain in specific detail how and why a motley assemblage of thousands of hackers working for free on their own time ("the bazaar") could produce better software programs than the expensive professional talent amassed by Microsoft and other software companies ("the cathedral"). The answer has much to do with tapping into the decentralized intelligence of a huge pool of personally motivated programmers via the Internet. Others have observed that the cost of developing any software that is fundamentally better than current software will require more resources than any single corporation can afford - but that global networks of the best programmers motivated by prestige and peer recognition can generate vastly superior software, as GNU/Linux demonstrates.[10]

Some of Raymond's trenchant observations:

> -Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
> -Every good work of software starts by scratching a developer's personal itch....The best hacks start out as personal solutions to the author's everyday problems, and spread because the problem turns out to be typical for a large class of users.
> -Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
> -Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone. Or less formally, "Given enough eyeballs, all bugs are shallow."

The analysis of the "bazaar" model of software development was so compelling that Netscape, under siege by Microsoft's Internet Explorer, decided in January 1998 to release the source code of its Communicator browser. Netscape's move was a radical notion at the time, motivated at least as much by business calculation as moral conviction. Within months, many other software companies began to recognize the power and technical superiority of open code software -- as well as the niche profit opportunities in a marketplace dominated by Microsoft. Such major players as Oracle, Corel and Informix announced that they would be porting their products to GNU/Linux. Others, such as Sun, have made the source code of their programs available in some fashion or another, but under modified licensing rules that seek to maintain some measure of proprietary control. It has been especially significant that IBM has announced its support of GNU/Linux and the Apache Group (the open code community that has made Apache the dominant web server software) and that it will ship a free, no-frills version of its database program, DB2, for GNU/Linux in spring 1999.

The growing interest of software vendors and even chip-makers like Intel in open code software promises to trigger significant upheaval in the computing/software industries as the proprietary world seeks out new ways to exploit, domesticate or co-exist with open code software. It is unclear how GNU/Linux and other open-code software will fare if proprietary developers actively seek to arrest or control its growth. On the other hand, Red Hat, Caldera, SuSE and many other companies see novel business models based on a growing and robust open code software "market."

In any case, open code software clearly represents a serious alternative to the traditional proprietary model of software development. Its appeal is not just that it is cheaper, more versatile, reliable and customizable software. Open code software represents a structural shift of power from sellers to users, and in that sense is one of the most liberating tools of media empowerment that individual citizens and the civic sector might ever imagine. Open code re-positions the terms of competition to a matrix of quality and utility, and diminishes the advantages to be gained from manipulations of the distribution apparatus, marketing schemes, restrictive licensing terms, and bundling deals with hardware makers.

Open code software is also inherently more suited to educational environments because its inner logic - the source code - can be directly manipulated by students. With its inner parts visible, users can choose to learn how the software works, and then share and develop that knowledge. Proprietary software, by contrast, is inherently "unknowable" because its inner architecture is a trade secret.

The future of the open code software movement may hinge upon what new terms of engagement emerge between the proprietary world and open code software developers. Can the integrity of the free software vision and the vitality of its online communities be maintained as the movement expands and perhaps morphs into a more diverse open source movement? Section II explores the obstacles that must be addressed, followed by an agenda that H20 proposes for fostering the continued growth of the new software movement.

II. FORCES THAT MIGHT DERAIL THE OPEN CODE VISION

Open code software just may be the elusive "killer app" that the computer business has sought for so long: a powerful, novel software that "changes everything." Yes, its actual presence among average computer users is still quite limited. But as the poet Mary Oliver might put it, "It's not the size, it's the surge." GNU/Linux is the only non-Microsoft operating system showing growth in market share - 212% growth in market share in 1998; it has 17.2% of the market for server operating systems, a nearly three-fold gain over the previous year.[11]

Apache, Perl and many other open code software programs are being embraced by major proprietary vendors as cornerstones for future software development. Even Microsoft concedes in internal strategic analyses leaked to Eric Raymond -- the "Halloween Memos" -- that GNU/Linux and other open code software "provide very dramatic evidence....that commercial quality can be achieved/exceeded by open source software projects."[12] All of this suggests that the new software movement stands at a new threshold. It has a sovereign vision, a superior product, a burgeoning cadre of supporters, growing investment, fresh attention within technology circles and the general press, and a hardy development process of proven effectiveness.

But can this movement sprout wings and fly? Open code software faces some formidable challenges if it is to grow and become a broader consumer phenomenon. Among the barriers it faces are the lack of popularly accessible documentation and technical support; the lack of a clear, well-known brand identity and marketing support; a susceptibility to code-forking that can vitiate the development process; the shortage of enterprise-oriented development tools; development uncertainties that make proprietary vendors queasy about porting applications to open-code operating systems; and problems with integrated, end-to-end ease of use. These issues raise questions about the ultimate strength and durability of the open code movement, especially if proprietary vendors seek to domesticate or divert its software in order to protect their existing products and revenue flow.

This section outlines some of the challenges that proponents of open code software may face, both affirmatively in expanding the quality and usage of their products as well as defensively in neutralizing monkey-wrenching by competitors. It is unclear which of the challenges enumerated below will actually materialize and which opportunities should have the highest priority. But if the new software movement is to flourish, these issues certainly deserve further inquiry and discussion.

A. Can the Integrity of the Open Code Vision be Maintained?

The dynamic tensions between the proprietary "cathedral" models of software development and the open code model are likely to intensify in coming months and years. Indeed, this very polarity may become more complicated as new proprietary/open code business models are developed. A key question is whether the open code movement can retain the integrity of its vision and community-driven development process as the proprietary world comes calling. Will established companies see open code software as a threat or promise, or both? There is no question that Microsoft sees open code software as a threat to its franchise. As the Halloween I memo astutely notes, the most significant threat is the open software development process: "The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization effort appears to scale." Halloween I concedes that open code software "poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS [open source software] has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat."[13]

Apart from such outside threats, however, there are questions about whether the open code community will retain its community vitality and mutual commitment as its popularity (and opportunities for entrepreneurial spinoffs) grows. Also, can open code communities think and act strategically? Vinod Valloppillil, the author of Halloween I, shrewdly points out:

> A very sublime problem which will affect full scale consumer adoption of OSS projects is the lack of strategic direction in the OSS development cycle. While incremental improvement of the current bag of features in an OSS is very credible, future features have no organizational commitment to guarantee their development.

It is an open question whether "organizational commitment" is needed to stabilize and guide open code software development. Given the radically decentralized nature of the new software movement, however, it could benefit from a "loose leadership" that helps convene relevant players, focus discussion, scout out the frontiers and develop strategic analyses. Such efforts are important if the movement is going to respond to the inevitable threats that the proprietary software world will contrive. There may be some lessons to be learned from the 1960s, in which leader-less political initiatives were sometimes out-maneuvered by the more strategically organized, resource-rich opposition. Without losing the moral legitimacy and political strength of its grassroots base, the new software movement needs to evolve new governance/leadership structures to deliberate and advance its interests.

At the same time, the open code vision will require a broader, more ecumenical engagement with computing professionals, especially within corporate MIS departments, and non-technical constituencies such as education, government, nonprofits and the general user. Maintaining a core vision while popularizing it is a signal challenge.

B. Can the Open Code Community Overcome Intellectual Property Snares?

Many open code advocates correctly worry that software companies will create copyrighted derivatives that will privatize the shared intellectual capital of the open code community - the very scenario that inspired Richard Stallman to create the GNU General Public License. This has happened, for example, with SendMail, the open-code mail transport software whose original author, Eric Allman, has turned it into a proprietary product (while promising to improve and support the open-code version). This trend may accelerate as more programmers decide to become entrepreneurs feeding off of open code networks.

While such adaptations of open code programs may seem relatively benign, similar tactics by a Microsoft using "embrace, extend and extinguish" tactics could be disastrous. Microsoft speaks of trying to "fold extended functionality into commodity protocols/services and create new protocols." What this means, Eric Raymond explains, is "introducing nonstandard extensions (or entire alternative protocols) which are then saturation-marketed as standards, even though they're closed, undocumented or just specified enough to create an illusion of openness." This, of course, is precisely the Microsoft strategy that a federal court enjoined recently in Sun's litigation. It charged Microsoft with "standards pollution" of Java, which is intended to be a "write once, run anywhere" software - a "virtual machine" that can run on a variety of operating systems.[14]

Clearly the GPL offers the best protection for expansion of open code software. But that may not be enough. Some distribution vendors combine non-GPL software with the GNU/Linux kernel and distribute them, introducing new copyright quandaries. Also Microsoft may be able to use its market power to prevent open code software from reaching a wider base of consumers. It could prohibit OEMs [original equipment manufacturers] from pre-installing GNU/Linux, for example, as it appears to be doing with Dell in Europe.[15] Even though Microsoft has recently allowed computer makers to take new liberties in configurations of their opening screens (allegedly in response to the DOJ lawsuit), Microsoft's underlying contract terms with OEMs have not changed.

Nor are any PC makers accepting a challenge by the chairman of Be Inc. to pre-load either BeOS (his company's operating system) or GNU/Linux for free - which would have obvious value to consumers.[16] None will do so, contends Jean-Louis Gassée, because it would violate Microsoft licensing provisions, which he says prohibit OEMs from modifying the boot manager to display non-Microsoft operating systems or allowing the display of "unapproved" icons on the desktop screen. "As an industry insider gently explained to me," said Gassée, "Microsoft abides by a very simple principle: No cracks in the wall. Otherwise, water will seep in and sooner or later the masonry will crumble."[17] If Microsoft's licensing terms could be resisted with impunity by OEMs, on the other hand, it could allow real competition among operating systems. It would also vastly expand the potential user base for open code software and moderate the prices for Microsoft products.

There are also questions about hybrid corporate-sponsored "noospheres" (discrete communities of open code programmers)[18] such as Mozilla, the Netscape-sponsored group of programmers dedicated to extending and customizing Netscape's browser. Now that Netscape is being bought by AOL, there are fears within the Mozilla community that AOL-Netscape could abandon Mozilla or push its work into an AOL-owned proprietary tree. AOL-Netscape, for its part, have their own fears of legal liability if the open-code process were to introduce patented code into the browser.[19] Similarly, Sun is trying to kick-start a community of open code developers for its Solaris software using a quasi-open code license which it calls a Community Source License.[20] The long-term viability of these experiments remains an open question.

C. Can Open Standards and Interoperability be Assured?

Beyond desktop and enterprise computing, the new software movement is committed to an open framework for the Internet. Fundamentally, this means open standards and interoperability among different hardware and software platforms. Open standards means that the standard-setting process is open to anyone; that technical specifications are published and freely available, and that no single vendor can control or manipulate the evolution of the standard. Ideally, there are certification tests to assure that an open technology actually meets the specifications.

The Halloween I memo suggests a chilling scenario by which Microsoft would subvert the commodity standards and infrastructure that is the foundation of the Internet by introducing proprietary protocols. The author suggests that Microsoft "de-commodize protocols and applications" by "extending these protocols and developing new protocols." This would undermine the core advantage of open code software, its ability to integrate diverse hardware and software through common commodity protocols. This is why Microsoft finds open code software in general and GNU/Linux in particular so alarming: it threatens to subvert its proprietary integration of systems - and thence its monopoly rents - with open, competitive integration based on user-determined standards of quality and utility. This is why Microsoft quietly changed its protocols to render Samba, an excellent open code communications application, incompatible with Microsoft's Windows NT and Windows 95/98 software.[21] This also explains why Microsoft is hostile to the IETF, the Internet Engineering Task Force, which has historically resisted the special pleadings of any single vendor in generating technical standards for the Internet.

When AT&T and the Bell Companies controlled the technical standards for telephonic network, no one else could compete. But once standardized interfaces were required to ensure interoperability, competition was possible - and innovation in all sorts of value-added products and services materialized. So it must be with the Internet, now and in the future. It is vital to ensure that the computer box, servers and switching protocols are interoperable, or else competition, innovation and their myriad benefits will be thwarted. The open code movement represents a powerful but under-mobilized constituency for assuring open standards and interoperability on the Internet.

D. Can Open Code Software Build a Brand Identity?

In our business-dominated culture, it is an inexorable reality that people respond to brands - an image and aura that distinguishes Brand x from Coca-Cola, McDonalds, General Motors and Microsoft. This may pose problems in expanding usage of open code software, because it has none of the contrived mystifications that come from advertising. It doesn't have an image, except perhaps its anti-image as a Birkenstock, counterculture artifact. While image may not matter much for techno-sophisticates, it could matter for the average consumer and even for the middle-managers of corporate MIS departments. Brand names are a form of cultural credibility. Even though that credibility may be unwarranted or unjustifiably expensive, it is, for now, a price of entry to the mainstream of American commerce and culture.

If it is going to expand beyond its grassroot niches and become an established player in mainstream computing, open code software may need to bring its image into focus and "manage" it with some strategic sophistication. Some argue that the modularity, technical superiority and increasing plug-and-play capacity of GNU/Linux may make traditional advertising gratuitous. This may prove true. On the other hand, the larger universe of open code software could become a perennial also-ran - an uncredentialed, second-choice alternative, despite its superior merits - unless it has some minimal branding and marketing support.

One small but important step in this direction is OpenSource.Org, a certification system started by Eric Raymond and others, which gives a seal of approval to open code software that meets ten specific standards.

Under the organization's criteria, the GNU GPL, Berkeley Software Distribution, and Mozilla Public License, among others, meet the standards.[22] (A similar open-standards certification for hardware may also be necessary as more software functionality become embedded in hardware interfaces, threatening the open protocols that allow competition and innovation.)

What really needs to occur is a more coordinated cooperative association of powerful brands. This could resemble coop ad campaigns for an entire product category -- milk, eggs, books - financed by a group of vendors. On a parallel track, there could be an aggressive public education program launched through popular magazines, the general press and constituency-specific periodicals (e.g., higher education, nonprofit trade associations). There are also online vehicles, conferences and other vernacular venues through which the open code "message" could be popularized in highly credible ways.

It is unclear whether the various new software communities might wish to cooperate and "go legit" through advertising and marketing. Certainly such initiatives would need their support. But resistance could be self-destructive because ultimately, the "counter-cultural" identities of GNU/Linux, Apache and other open code projects are vulnerable to appropriation and caricature by competitors if there is no organized, strategic plan for "managing the identity" of the new software movement. As the Halloween memos frankly admit, this is a battle for "mindshare." Part of the success of open code products will require the savvy, pro-active cultivation of mainstream awareness and acceptance.

E. Can Open Code Software Be Made User-Friendly?

While GNU/Linux has achieved remarkable growth among the technically proficient, its future among the average computer user will require improvements in user-friendliness. The modular architecture of GNU/Linux, which makes perfect sense for the technically proficient, may prove troublesome for the mass-market consumer. Generally speaking, open code software has not undergone extensive testing or modification with non-techies to make them easier to use. They are, after all, products of the hacker community. Microsoft has built much of its empire on the seamless integration of a user-friendly operating system and applications: a value-added monopoly function, poorly achieved, for which it charges a significant premium.

The parallels with the pre-divestiture AT&T are striking. Before the 1984 AT&T consent decree, there was no competition; just a regulated monopoly. In the years after that watershed, a flourishing telecom marketplace rife with innovation emerged. We stand at a similar pre-divestiture juncture with Microsoft, in which competition in the operating system market and other ancillary applications markets is artificially limited. A large, robust marketplace of modular, open-code applications stands ready to emerge, but has not. There is hope on the horizon, as Red Hat and other vendors begin to offer integrated packages and components, and as new desktop environments such as KDE and GNOME gain in popularity. Even more striking, IBM is now planning to offer round-the-clock tech support and service for Red Hat Linux. All these steps will help "grow" the market for GNU/Linux and open code software. But we have a long way to go before such "end-to-end" ease of use will be achieved.

F. Can Open Code Software Attract Support from Non-Technical Constituencies?

Ralph Nader became an early advocate of open code software because it radically empowers consumers in many ways. The benefits are not undifferentiated benefits that flow to the aggregate consumer, but highly individualized benefits that may help particular niches of consumers: education, nonprofits, voluntary civic associations, charitable endeavors, professional associations, and business users as well. As open code operating systems become more pervasive, it becomes more plausible to develop more stable, enduring open code platforms and compatible applications. The question is, can non-technical constituencies be made aware of their enormous stake in the emergence of open code software and help spur a demand-pull for it?

Since much of the actual, long-term value of open code software is not immediately understood by the layperson, this could require a lot of missionary work. Indeed, in the short term, open code software is likely to be far too technical and complex for the average user to find comfortable. The systems must evolve more. But this should not deter new initiatives to educate non-technical constituencies about how open code software represents a community- and knowledge-building infrastructure without precedent. It represents a sovereign "media space" independent of proprietary manipulations, arguably one of the most powerful potential forms of user-empowerments in electronic media.

III. STRATEGIES FOR SUPPORTING OPEN CODE SOFTWARE: THE H20 AGENDA

As the preceding analysis suggests, the new software movement is poised to take off and reshape the terms of competition in the software marketplace (and much else besides). But to do so, it must overcome a variety of challenges from proprietary competitors and develop its own capacities as an unconventional software development and distribution system. No one can or should dictate to this community what it should do; its indispensable strength is its participants' sense of ownership and support. Yet new leadership from an independent, movement-owned organization could help consolidate diffused energies, accelerate the adoption of existing open-source software, pro-actively develop new strategic directions, and reach out to a broader array of computer users, legislators and the public.

H20 proposes to play such a role through a number of interrelated initiatives. The Project will provide a useful focal point for research, discussion, policy analysis and strategic planning for the new software movement. It will be more feasible to air and debate disparate perspectives within this community and develop new syntheses, consensus and action plans.

H20 will reach out to non-technical constituencies to inform them of their stake in the development of open code software and develop their own projects based around it. Initially, the Project's focus will be on educational applications of open code software, but we also envision outreach to nonprofit organizations, civic groups, voluntary associations, professional societies, government agencies, and small businesses, among other groups.

Besides some prominent leaders of open-source projects, H20 will have the backing of the Harvard Law School's Berkman Center on Internet & Society, whose six faculty members[23] , eighteen fellows and eighteen affiliated individuals represent a diverse network of professors, students, entrepreneurs, lawyers and "virtual architects." The Center's faculty members regularly teach courses of major issues facing the Internet. It has hosted such major conferences as the Internet & Society Conference in May, 1998 and the November 1998 conference of ICANN, the Internet Corporation for Assigned Names and Numbers, which is in the process of becoming a new governing entity for the Internet.

A. ARTICULATING THE PUBLIC'S STAKE IN SOFTWARE AND THE INTERNET

As open code software gains in popularity, it will face any number of significant challenges that will require sophisticated analysis based on timely empirical knowledge. This section describes some of the key intellectual projects that H20 will facilitate to help expand the new software movement.

1. Develop richer theoretical and strategic analyses to explain the power of open code software development.

One of the most important challenges facing the open code software movement is to refine and deepen its strategic intelligence. It is paradoxical and disturbing, for example, that two of the most thorough, timely and hard-nosed analyses of the open code software movement - the Halloween documents - were generated by Microsoft. To be sure, there are a fair number of academics and computing iconoclasts who are highly informed, capable analysts. But this work is diffused and needs greater sustaining support.

The work of open code opinion-leaders would be enhanced if they had more opportunities to meet, converse informally, plan strategically, develop personal relationships, cultivate new institutional collaborations, organize to address common goals, and communicate them with larger constituencies and the general public. The kinds of interpretive syntheses prepared by the Raymonds, Stallmans, Torvalds, Lessigs and Barlows need to be supported, showcased and brought to the attention of mainstream culture. H20 aspires to offer such support to the new software movement.

Raymond, Stallman, Valloppillil and others have offered important conceptual critiques of the open code software process, but these are really opening skirmishes in a much larger, ongoing intellectual inquiry. There is a need to probe the strengths and limitations of open code software development, and to devise new theoretical tools for understanding it. Received theories of intellectual property and economics, for example, simply cannot explain the "gift cultures" represented by open code communities. Nor can they explain why the weak intellectual property protection of the HTML protocols for the World Wide Web utterly demolished the closed, proprietary approaches by AOL and CompuServe, producing one of the most phenomenal publishing successes in history.

The proprietary world clings to archaic economic theories and legal protections despite the onslaught of a new computing paradigm, even as the new paradigm, awakening to its actual powers, struggles in fits and starts to understand itself. For example, information industries objected so strenuously when government agencies began placing so much taxpayer-generated information on websites, for free. Yet the dire predictions proved false: information vendors have migrated into new lines of value-added information synthesis that complement the government's free disclosure of information. The government stimulated the creation of new niche markets and innovation while shutting down a subsidized, socially wasteful market whose functions could be better and more cheaply served by government itself.

We need more analysis of market transformations that have public implications so that the public and policymakers can make more informed, independent assessments. This would include new economic theories about information technologies and products; new antitrust theories that take account of the novel tactics that can stymie competition; and new analyses of how interoperability in hardware is becoming more critical as software functionality migrates to proprietary hardware interfaces.

To be sure, information theorists are developing new approaches, exemplified by Shapiro and Varian's Information Rules: A Strategic Guide to the Network Economy.[24] But the new theorists do not necessarily regard consumers, democracy, education or the public interest as their foremost concerns. Nor do many of the emerging theories seek to expand public interest values into new realms. Why, for example, shouldn't the principles of open standards be applied to the desktop computing environment, with the commoditizing of the operating system? The open code software movement needs this kind of serious strategic intelligence capacity.

2. Explain the democratic, civic and consumer implications of open code software.

Although the new software movement is gaining visibility, there is relatively little understanding of how open code software empowers consumers and citizens and why it is important to our democratic culture. New analysis and public education are needed to explain the pro-consumer benefits of open code software, both in comparison to existing proprietary products (quality, reliability, flexibility, etc.) as well as in neutralizing the anti-competitive behavior of proprietary software makers, especially Microsoft. This analysis needs to be developed and popularized so that the latent constituencies with a stake in open code software - nonprofits, government, education, etc. - can begin to understand their long-term interests and act accordingly.
The public sector in particular needs to understand how its considerable purchasing power could help "grow" this market and expand its benefits; how open code software could provide the scaffolding for a new "media space" for the civic and education sectors; and how software design is a legitimate realm of public concern.

3. Develop a new taxonomy of intellectual property and software development models.

Although the General Public License (GPL) and Berkeley Software Distribution (BSD) license are popular licensing models, there are a wide variety of open code software licenses in use, many of which blend arcane proprietary goals with open-source development freedoms. It would be useful to develop a detailed taxonomy of this confusing legal domain so that the major patterns of development and their legal and market implications could be better understood. For example, which licenses are compatible with each other? Could a browser that supported commercial technologies (like Java, RealAudio and any number of plug-ins) be protected under GPL? Will the implementation of Harmony as an alternative to Qt succeed, and will the non-GPL open-source licensing model for Qt succeed ?

On a more ambitious level, it would be helpful to fortify the concepts of copyleft with stronger standards under federal copyright law. Why shouldn't the law validate and recognize the GPL and perhaps offer extra civil and criminal sanctions for those who seek to deform its provisions? As an artifact of another era, copyright does not recognize alternative incentives for artistic and creative expression, the social recognition and prestige that occurs in gift-culture communities. It would be useful to develop new legal doctrines that might adequately recognize and protect these motives so that they might have a modicum of legal standing.

4. Examine the threats to open standards and interoperability in the Internet and other computing systems.

The fundamental strength of the Internet as a communications tool and marketplace is its open standards and protocols. Everyone can theoretically compete and communicate without artificial gatekeepers. The various bodies that have given rise to the Internet and assisted with its governance - ARPANET, IETF, IANA (now ICANN) -- have all honored openness and consensus. It is important that this tradition continue and that no "public space" or node of the Internet be controlled by a single vendor or oligopoly of vendors. Yet the threat of a balkanization of the Internet's open standards is growing, as exemplified by the receding government role in supervising the Internet and in the laissez-faire policies proposed by the Department of Commerce and former White House advisor Ira Magaziner.

Ensuring open standards and interoperability is exceedingly difficult when so many of the issues are highly technical and the industry/Internet decisionmaking bodies are not obliged to follow the standards of openness, citizen access and due process that a government body must observe.[25] Even if open access were legally required, the user community cannot readily afford to participate on an equal footing with industry representatives, with all the travel, sustained research, writing and networking that meaningful participation entails. This lamentable situation means that software companies can sometimes "lock up" software protocols that rightfully should be subject to open discussion, review and perhaps modification.[26]

H20 and its network of friends would seek to fill this void by participating in standards deliberations, preparing important technical analyses from the public interest perspective, and alerting the larger public to their stake in the process. It would be a key priority of H20 to monitor the status of open standards and interoperability, identify the primary threats to those goals, and propose steps to ensure that no one captures the key public spaces or interfaces of the Internet.

5. Identify barriers to market entry for free software operating systems.

Under its consent order with the Justice Department, Microsoft cannot prohibit OEMs (original equipment manufacturers) from selling other operating systems as pre-installed software. But that agreement only covers Windows 95, not Windows NT, and applies only in the United States. There are allegations that Microsoft's contract with an OEM in Europe prohibits it from advertising or marketing other operating systems, such as GNU/Linux.[27] The Halloween II document suggested that Microsoft could reduce the price of its NT software, if needed, to pressure OEMs not to pre-install GNU/Linux.

This is only the most egregious known example of how the proprietary world might seek to impede the growth of open code software. It would be useful to explore what other discriminatory barriers to entry may exist and bring them to the attention of Justice Department officials and the computing public. Neither Microsoft nor other companies should be allowed to use their market power to squelch the marketing or use of open code software alternatives.

6. Explore alternative methods of financing the development of free software.

The free market/libertarian ideology that prevails in high-tech circles reflexively dismisses the idea of government playing a salutary role in the marketplace. But, of course, government support was indispensable for the development of the Internet (via ARPANET) and the Netscape browser (derived from the government-funded MOSAIC software). It may not be coincidental that as public sector involvement in the Internet and software declines, the incidence of anti-social, anti-consumer activities (e.g., industry consolidation leading to less competition and choice; Microsoft's anticompetitive tactics; bolder attempts to privatize open standards, etc.) seem to be increasing.

A strong case can be made that government-mediated support for software and networking innovations has an invigorating effect on the larger marketplace. There is no reason why government should not have a hand in actively promoting open code software as an augmentation/pacesetter to the commercial sector's products. The impact on market competition, innovation and consumer well-being is simply too great.

Many feasible schemes are imaginable. James Love of the Consumer Project on Technology, the Nader-affiliated group, has proposed a 1-2% vendor tax on commercial software that would be used to finance the development of GPL-licensed open code software at state universities. Internet innovator Carl Malamud has proposed a tax credit scheme for the same ends. Students could receive work-study support; the software marketplace could be infused with fresh ideas and new programming talent; and the public would get free software and new tools for combating anticompetitive practices in the marketplace. Other financing schemes could include funding by consortia of industry, universities or Internet service providers, which might find it useful to support infrastructure costs to help keep GNU/Linux and other open code software competitive. All these parties, after all, benefit from the open code software that currently makes the Internet function so well.

7. Advocate greater public access to government information.

It is a truism that information is the currency of democracy. Yet even as the Internet is becoming a common tool in the conduct of everyday affairs, many parts of the federal government - and state governments as well - continue to resist putting useful information resources online. The failure to do so is sometimes an attempt to stifle public scrutiny and participation; other times information is simply privatized, unnecessarily enriching private vendors at the expense of taxpayers while diminishing democratic accountability.

Several areas deserve attention. The NTIS, National Technical Information Service, has become a multi-million subsidy to information vendors at the expense of ordinary citizens who cannot afford to buy NTIS documents or data tapes created by their own tax dollars. Releasing this information would not only be the fair thing to do, it would invigorate scientific and technical debate on issues that are otherwise dominated by well-heeled businesses and institutions. Court and administrative agency transcripts are another body of government information that ought to be put online. Currently, however, this information is owned by the official transcribers, preventing citizens from following court and federal agency proceedings that they are financing, many of which are of great public interest.

Many congressional records, too, such as committee reports and legislative mark-ups, are not available on the Internet, notwithstanding the speed with which Congress put the Starr report on the Internet. Finally, Freedom of Information Act requests that do not involve privacy concerns could and should be placed on the Internet. Not only would this make vast amounts of the public's information readily available, it would save the government considerable sums by eliminating costs of processing duplicate FOIA requests.

B. CREATING A SOFTWARE USERS' ALLIANCE

It is important that open code software developers and users have hospitable forums in which to discuss common concerns. They also ought to have the capacity to organize themselves to assert their strategic, legal and political interests as they are affected in the marketplace, industry decisions, public policy and elsewhere. Section B outlines several strategies by which H20 could foster these goals.

1. Convene programmers to help develop new open code programs for niche constituencies.

Open code software programmers do not necessarily need any guidance from H20. On the other hand, people who create open source code software do not necessarily consider novel, transforming applications of their talents. The leadership of a Torvalds or Stallman or Allman is needed to propose a vision and nourish the implementation. H20 could play a valuable role in convening developers to contemplate larger, more ambitious applications of their expertise for students, educators, people of ethnic heritage or narrow affinity groups.

H20 envisions a role of catalyzing educational, business, governmental, and cultural institutions to explore how open code software can help them. Small investments by a number of individual leading institutions can help develop a common, cost-efficient software platform that serves the needs of the Internet community at large.

2. Revamp government and education purchasing to support open code software.

It is not widely appreciated how government, using its own immense market power as a consumer, can dramatically affect the levels of innovation, quality and price in the marketplace. Federal, state and local governments spend more than 18% of the nation's GNP, representing a powerful, largely untapped force for shaping the quality of products delivered. This can be seen in how federal procurement finally persuaded automakers to bring air bags to market. It can also be seen in government-led expansion of markets for recyclable products, chlorine-free paper, longer-lasting road pavement and countless other innovations. Nader, a champion of the strategic uses of government purchasing power, explains: "If properly deployed, government's massive pool of organized purchasing power can jump-start the adoption of new or dormant products and technologies to promote higher productivity, greater human safety and a cleaner environment."[28]

Government agencies typically buy off-the-shelf software products instead of issuing detailed performance specifications to would-be vendors. Some agencies actually specify the vendor - i.e., Microsoft - rather than the generic functions that the software must meet. The federal government's short-term approach to its routine software purchasing means that government wastes lots of money on deficient proprietary software rather than using its imagination to reap better open code software that will not be subject to costly proprietary restrictions and upgrades. Governments in developing countries are starting to realize that the licensing fees of U.S. software vendors are simply too expensive (especially as trade agreements are enforced more vigorously) when there are cheaper, technically superior alternatives such as GNU/Linux. Mexico is reportedly shifting more than 140,000 computers in educational settings to GNU/Linux for this very reason.

Higher education, also, has much too much leverage - student-consumers, programming talent, institutional purchasing dollars - to accept the limited choices offered by proprietary vendors when open code software offers greater long-term cost savings, flexibility and stability. Instead of flexing their muscles in this direction, however, many major universities allow themselves to serve as marketing and recruitment vehicles for Microsoft, accepting discount software deals in return for exclusive access to a future consumer base and programmers' mindshare. But even these seemingly smart deals can prove to be more expensive over the long term as licensing regimes are changed to extract higher prices once the company has a monopoly stranglehold.[29] It is entirely appropriate and consistent with higher education's public mission and taxpayer expenditures that the software it supports be open code software. New initiatives in this area could yield stunning benefits to higher education over time.

For those who object that government has no business short-circuiting the marketplace or the verdict of open code communities, the answer is that government has to buy something. Why shouldn't it be the best-performing open code software, as determined by performance specs and open competition? The case for more intelligent, innovation-minded government purchasing -- rather than the slavish acceptance of brand-names - is compelling. But first, the case must be made in a more detailed way and presented to Vice President Gore, the GSA Administrator and the public.[30] If successful, even in small instances, government purchasing of open code software could have powerful psychological and economic repercussions as smaller purchasers follow suit and expand the open code community/market further.

3. Assert consumer rights in the software & Internet marketplace.

While individual users enjoy remarkable new capabilities through the Internet, their ability to assert their common strategic interests in the new electronic marketplace and public policy issues is limited. Users have few forums or resources that can effectively advocate their collective interests in consumer rights, intellectual property, online privacy, access to government information, interoperability of computer systems, among other issues. H20 plans to be a convenor and supporter of diverse efforts to promote consumer rights in the software/computing/Internet environment.

The need for sophisticated legal advocacy in this area is exemplified by the little-known revisions being prepared for Article 2B of the Uniform Commercial Code, a cornerstone of American business law that governs the sale and lease of software, databases and information. The American Law Institute and the National Conference of Commissioners on Uniform State Laws are redrafting this provision of the law which, if adopted by all 50 states as intended, would become controlling law for transactions constituting 30% or more of the U.S. economy. The redrafting of the law is dominated by industry lobbyists, leading to the not-surprising result that the new draft language will allow sellers to legally bind consumers to surprising, unreasonable legal provisions buried in pages of legalese so long as the user clicks the "I agree" box on a software program or web site. As it happens, this proceeding is being monitored by Harvard law professor Lawrence Lessig[31] and by Todd Paglia of the Consumer Project on Technology.[32]

Smart, sophisticated user representation in obscure, industry-dominated arenas remains fairly rare. Yet as the call for industry self-regulation grows - in standard-setting, international trade, consumer protection, privacy, intellectual property law, and many other realms -- there are serious questions whether governance of the Internet will become privatized and our democratic traditions abandoned. It is vital that user/citizen interests be asserted more forcefully and intelligently so that governance can be truly responsive to sovereign user interests and not just business interests.[33]

4. Develop and "manage" a brand identity for open code software.

As discussed in Section II, the new software movement must begin to develop a brand identity if it hopes to expand its influence in the mainstream computing environment and enlist the support of non-technical constituencies. This need not imply competing with the proprietary world's marketing budgets and advertising reach - a style of battle that the open code movement can never win. Indeed, the open code movement's grassroots identity and gift-culture ethos call for a different style of marketing. Yet marketing, nonetheless, is needed. It is presumptuous and premature to say what such a campaign might consist of. But H20 does hope to convene leading actors in the open code world to facilitate a needed discussions about this topic.

Part of projecting a brand identity is knowing what open code products are available and helping users learn about them and acquire them. To this end, H20 hopes to facilitate the creation of a new clearinghouse for open code enthusiasts. There are hundreds of open-source programming communities for hundreds of different purposes. Assembling a more comprehensive inventory of these projects and making them more readily available would have an immense catalytic influence.

5. Serve as a repository and users' forum to sustain and improve abandoned software programs.

As the pace of innovation in computers and software accelerates - and as corporate mergers and acquisitions reconfigure the marketplace - many vendors are abandoning popular and entirely effective software programs. Popular programs such as Sidekick, Xtree, EchoPro and soon, it is predicted, Eudora, are being thrown on the dustbin, forcing satisfied users to buy new software. H20 proposes becoming a standing repository for these programs by becoming the legal owner of the source code and, where possible, helping interested user groups to organize themselves to sustain legacy products. The vendors donating the software could receive tax deductions for their beneficience, and the software could be put into the public realm through a General Public License. Interested users could keep alive programs that they like, adapt them to new hardware systems and improve them in other ways.

Having H20 serve as a repository for legacy software could also be of immense value to libraries and archives. "Digital files are utterly brittle," writes futurist Stewart Brand. "They're complexly immersed in a temporary collusion of a certain version of a certain application running on a certain version of a certain operating system in a certain generation of a certain box, and kept on a certain passing medium such as a five-inch floppy."[34] Now that the half-life of new computer technology is about three to five years, professional librarians and archivists are facing the practical consequences of relentless technological innovation: millions of documents cannot be easily accessed. By providing a practical means for old software systems to continue to function, H20 would help meet the formidable challenges of document preservation and access.

CONCLUSION

Today, at the dawn of the Internet as a popular medium, as major corporations vie for hegemony over the new communications infrastructure and standards, open code software holds great promise to forge a new, more socially constructive path. It offers the chance to empower consumers, nonprofits, education and various civic segments of American society. It allows for more open, flexible architectural designs for markets, communities and cultures. It enables the construction of cheaper, more durable new media platforms which can accommodate scalability and innovation with great ease.

Capitalizing on this historic opportunity requires the commitment and support of the nation's foundations, one of the few American institutions with the resources, stature and independence to provide risk-taking leadership at this time. This proposal is an attempt to open a dialogue and to make the case for founding a new organization, H20, to facilitate the growth of the open code software movement.

We are mindful that not all moments in history are alike. Lost opportunities are not easily recovered. American life might have evolved along a very different path in the 20th Century if nonprofit advocates had won the debate between 1928 and 1935 over who would control the public airwaves.[35] Commercial broadcasters prevailed, of course, and citizens, educators, labor, religious groups, artists and untold other Americans were largely banished from radio and television. The medium came to be identified with a narrow range of (commercial) formats and untold alternative possibilities were simply never developed.

Unlike the struggle to control broadcasting, the coming struggles over software design, Internet governance and the conduct of electronic commerce will not be resolved through public policy alone. Outcomes will also pivot upon the ability of software users and developers to assert their sovereign interests in the marketplace, industry fora, standards-setting bodies, legal venues, the trade and general press and other vectors of influence. In this early stage of the Internet and software design, as critical decisions are made, it is important that we strive to take risks on the best future we can imagine if only because there may be highly unpleasant, irrevocable costs for not doing so.

David Bollier
March 10, 1999

David Bollier is an independent journalist and consultant based in Amherst, Massachusetts, who writes frequently about the social and democratic implications of the electronic media. A student of citizen advocacy, progressive politics and cultural change, he has worked for the past fifteen years with television writer/producer Norman Lear on assorted non-television projects; with Ralph Nader on a number of civic empowerment/public policy initiatives; and with several foundations and activists who know how to grow new vectors of possibility. In recent years, Bollier has proposed a strategic agenda for reinventing democratic culture using new electronic media (http://www.netaction.org/bollier/index.html); chronicled socially visionary models of business management in his book Aiming Higher; and synthesized the economic, social and political reasons for curbing suburban sprawl and rejuvenating urban regions (http://www.sprawlwatch.org). Bollier can be reached at bollier@essential.org.

NOTES

1.
Much of this debate revolves around what kinds of software licensing and distribution ought to prevail, and what terms of engagement with the proprietary software world should be entertained. Richard Stallman, a leading software development, prefers the term "free software" to stress the freedom to use source code as one sees fit and the freedom to use derivative programs as well, without the customary restrictions that copyright law imposes. Others, such as Eric Raymond, prefer the term "open source software" to stress the accessibility of the source code even if proprietary derivative versions (having closed, copyrighted source code) are allowed. It should be noted that "freeware" and "shareware" are not appropriate synonyms for "free software" because "freeware" does not necessarily have open source code; it can simply be a free or promotional version of a program whose binaries (but not source code) are accessible to users. A review of the many strands of the new software movement can be found in Charis DiBona, Sam Ockman & Mark Stone, editors, Open Sources: Voices from the Open Source Revolution (O"Reilly & Associates, 1999).

2. See "Only the Free World Can Stand Up to Microsoft" at http://www.contex.com/ftwalk/FreeWorld.html, a site that is currently inoperative but seeking a new home. Hull's critique echoes Ralph Nader's seminal essay, "How to Think About the American Economy," The New York Review of Books, September 2, 1971.

3. A few noteworthy examples: Sendmail is the program that routes over 80% of all email on the Internet. Perl is the programming language that allows dynamic features on many web sites. Apache is the most popular web server software in use on the Internet. BIND, the Berkeley Internet Name Daemon, is the de facto DNS server for the Internet. Mozilla is the open code software used in the popular Netscape browser.

4. See, e.g., David Bornstein, The Price of a Dream: The Story of the Grameen Bank (University of Chicago Press, 1997).

5. See http://www.timedollars.org, and Edgar Cahn and Jonathan Rowe, Time Dollars: The Hidden Currency That Enables Americans to Turn Their Hidden Resource - Time - Into Personal Security and Community Renewal (Rodale, 1992).

6. For more on the Free Software Foundation and GNU Project, see http://www.gnu.ai.mit.edu/fsf/fsf.html.

7. Two excellent profiles of Linus Torvalds and the evolution of GNU/Linux are: Glyn Moody, "The Greatest OS that (N)ever Was," Wired, Issue 5.08, August 1997 or http://www.wired.com/wired/5.08/linux.html), and Josh McHugh, "For the Love of Hacking," Forbes, August 10, 1998. Linux International, a nonprofit organization formed to promote GNU/Linux use, can be reached at http://www.li.org.

8. Tim O'Reilly, "The Open Source Revolution," Release 1.0, November 1998.

9. The latest version of Raymond's essay can be found at http://www.tuxedo.org/~esr/writings/cathedral-bazaar.

10. See, e.g., Robert X. Cringely, Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition and Still Can't Get a Date (New York: Harperbusiness, 1996).

11. The number of copies shipped to customers more than tripled from 1997 to 1998, fueled by anti-Microsoft sentiment, strong performance and low pricing, according to a study by International Data Corporation. See story by Stephen Stankland, CNET News, December 16, 1998,
http://www.news.com/News/Item/0,4,30027,00.html?st.ne.ni.rel. Also, The Economist, February 20, 1999, p. 63..

12. See http://www.opensource.org/halloween1.html.

13. See http://www.opensource.org/halloween1.html.

14. Steve Lohr, "The Microsoft Trial Reviews Dispute with Sun Over Java," The New York Times, December 11, 1998.

15. This is based on accounts posted on http://www.slashdot.org by John Bryan, among others.

16. See John G. Spooner, PC Week Online, February 22, 1999, at http://www.zdnet.com/pcweek/stories/news/0,4153,1013959,00.html. /li

17. Ibid.

18. The term, as used in this context, was popularized by Eric Raymond in his essay, "Homesteading on the Noosphere," at http://sagan.earthspace.net/~esr/writings/homesteading/.

19. See, e.g., Ben Elgin, "Netscape to Cut Mozilla's Cord?" in Sm@rt Reseller, at http://www.zdnet.com/zdnn/stories/news/0,4586,2176488,00.html.

20. See Deborah Gage, "Sun (Almost) Opens Solaris," in Sm@rt Reseller, at http://www.zdnet.com/zdnn/

21. stories/news/0,4586,2215357,00.html. /li

22. See Robert X. Cringely, "The Pulpit," at http://www.pbs.org/cringely/pulpit/pulpit19980924.html.

23. See http://www.opensource.org/osd.html.

24. Charles Nesson, Lawrence Lessig, Jonathan Zittrain, Arthur Miller, Charles Ogletree and Terry Fisher.

25. Carl Shapiro and Hal R. Varian, Information Rules: A Strategic Guide to the Network Economy (Boston: Harvard Business School Press, 1998).

26. These include the rules set forth by the Administrative Procedures Act, the Federal Advisory Committee Act and the Government in the Sunshine Act.

27. An excellent example of this syndrome occurred in January 1999 when Microsoft was awarded a patent covering the use of style sheets in electronic publishing. With no clear notification to the W3C community or the wider computing world, Microsoft succeeded in essentially privatizing standards that resemble those used in the World Wide Web Consortium's Cascading Style Sheets (CSS) and eXtensible Style Language (XSL) standards. See http://www.camworld.com/misc/mscsspatent.txt or Antone Gonsalves, "Questions Raised About Microsoft Patent," PC Week Online, at http://www.zdnet.com/pcweek/stories/news/0,4153,1013845,00.html /li

28. See Letter to the Department of Justice by the Consumer Project on Technology, June 15, 1998, at http://essential.org/antitrust/ms/jkjun151998.html.

29. Ralph Nader, Eleanor J. Lewis and Eric Weltman, "Shopping for Innovation: The Government as Smart Consumer," The American Prospect, Fall 1992, pp. 71-78. Nader's Government Purchasing Project is a leading agitator for government procurement to promote environmentally sensitive and energy-efficient technologies. See http://www.gpp.org.

30. See Nathan Newman's report, "Microsoft Goes to College" in the October 4, 1998, online issue of NetAction's M$ Monitor, at http://www.netaction.org.

31. See the NetAction report by Mitch Stoltz, "The Case for Government Promotion of Open Source Software," at http://www.netaction.org/opensrc/oss-report.html.

32. See http://www.thestandard.com/articles/display/0,1449,2583,00.html?01.

33. See http://www.cptech.org/ucc.

34. An excellent critique of this issue is made by Lawrence Lessig in "Governance," a keynote address to the CPSR Conference on Internet Governance, October 10, 1998, at http://www.cpsr.org/conferences/annmtg98/.

35. See, e.g., Robert W. McChesney, Telecommunications, Mass Media and Democracy: The Battle for Control of U.S. Broadcasting, 1928-1935 (New York: Oxford University Press, 1993).

36. See, e.g., Robert W. McChesney, Telecommunications, Mass Media and Democracy: The Battle for Control of U.S. Broadcasting, 1928-1935 (New York: Oxford University Press, 1993).