Open Source - Infrastructure for Democracy

In a world where we must cope with terrorism and disease, software sometimes seems trivial. When our civil liberties are in danger, how important is source code? Are the rights assured by the GNU General Public License important when people are starving?

To understand the role of software in our complex society, we must investigate the role of mass communication in a democracy, and the way that computer networking may help to democratize mass communication, and thus lead us to better and more democratic societies.

It shouldn't be a surprise that a non-democratic regime instigated the violence on 9/11 . It's more of a surprise to view the failure of the democratic process that has proceeded in the purportedly democratic nations now involved in that war. Most obvious was the failure of the mass media in the United States and elsewhere in their duty to analyze and criticise. Whether a war is justified or not, a television network's job should be to provoke thought, not cheerleading, among their viewers. Had they been fulfilling that role, and had they been presenting to the American people before 9/11 what other cultures thought of Americans, things might have been different. This gives us a measure of the importance of the mass media.

The dialogue between a people and their government depends on mass communication. Until the advent of inexpensive mass printing based on movable type, heralded by Motoki Shozo in the East and Gutenberg in the West, the world's only democratic councils were themselves elites. Politics was the exclusive domain of the wealthy and well-connected, the only people who could afford to come together in person regularly to discuss politics. Until the printing press, news and craft were communicated mostly by voice, and the common person had little idea of what was going on farther away than the horizon. How much could that person participate in governing his nation?

The printing press has been superceded by the broadcast media, the second generation of mass media. The third generation, of course, is the internet. Many individuals already use the web as their primary source of news.

Let's talk about that second genration: the broadcast media. Mass communication is key to democracy, and today's mass communication depends on scarce and expensive resources, such as radio spectrum, that are administered by the government or are the effective property of a wealthy few. Their control of our mass communications gives that wealthy few too much power over what voters are told. Voters will cast their votes on the information they've been given, accurate or not.

This is why we should be concerned about the web and internet. The web is the most important tool for democracy since Shozo and Gutenberg. And this is why the web, and the software that enables it, are a treasure to be guarded and protected by any true democracy.

With a web site, anyone is a broadcaster. The only scarce resource is the attention of web readers, and of course some of the web's most popular sites started very small. If our societies are to be democratic ones in the future, internet-based discussion will be the main way in which we carry out the dialogue between people and their government.

But can we maintain the internet as a channel for democracy, or will it too become the property of a wealthy few? Will the individual still be able to participate as much as he can today? There are a few clouds on that horizon. Monopolies are the enemy of economic competition and democracy. And it's possible for a few monopolies to gain control of the internet to a much greater extent than today. To protect democracy and capitalism, we must make sure that control of the internet remains in the hands of the many, rather than the few. No company should have a monopoly on internet software, internet connections, or internet content. These areas need to be maintained as level playing fields for competition.

Let's think about the treasure that we have in today's internet, the part that could so easily be destroyed. Today, with simple web tools, everyone can find a community of people across the world who share their ideals. There are no more isolated people with crazy ideas, because every crazy man can find 50 other crazy people just like him on the internet! And 50 people can get a lot done! So, let's take as our example of crazy people who met over the internet: the Free Software developers. Their ambition, elucidated by Richard Stallman in 1984, was to build a world-class operating system and tools. In their spare time.

Without much money. Without being paid.

Operating systems, until then, had been the province of very large companies, and these Free Software nut-cases proposed to do it in their spare time, or at school, or as a low-priority task at work, and they would give it away to all comers, and allow you to copy it without cost. Obviously, the had no hope of success, right? But those 50 crazy people became 50,000. And they changed the world. Why did such a different idea succeed?

It was the power of the Internet that made widespread Open Source development practical. The Internet made it practical for a software developer in Japan to carry out a close collaboration with another developer in France or Africa, and for people with very limited resources to combine them and come up with enough power to get the job done. Open Source development was probably an optimal task for the early internet. Programmers are often very comfortable with textual communication, and thus they were happy to use simple e-mail lists for their communication, and FTP for their file exchanges. They were happy without web browsers. But they weren't content to stay that way.

Steve Jobs' office was down the hall from mine at Pixar. When I left there in 1999, to start working full time on Free Software, I dropped in on Steve and asked what he thought about Linux. Steve didn't think it would go anywhere, because he didn't believe we'd be able to make a good graphical user interface.

"I've had a lot to do with two of the three great GUIs", Steve said. I guess he meant the Mac Desktop and NexTStep... and the other one must be Windows. And Steve went on: "it took a billion dollar research organization to make every one of them". So, Steve didn't think that Free Software would meet the challenge of producing a good GUI. And even in the Free Software camp there was doubt about that, and the argument about it resulted in our making two good GUIs rather than one. Four years later, Steve Jobs went on stage at MacWorld to announce that the Macintosh web browser would be derived from one produced by the KDE Open Source GUI project, under the GPL license.

So, we know that Open Source can be surprisingly effective for an unconventional idea. But what about its strange economics? The different economic paradigm of Open Source has been critical to its success. Many companies have tried to compete with Microsoft, and most have failed. But today, GNU/Linux is Enemy Number One in Redmond, Washington. Why has Free Software succeeded in doing what IBM, Sun, and Apple couldn't do with their own competing operating systems? Unlike the others, we don't try to compete with Microsoft on Microsoft's own terms. The difference in our economic paradigm, with its greater emphasis on collaboration and the customer, is the reason for our success.

But does Open Source mean we all have to give up profit, don tie-dyed shirts, and live in a commune? No. Open Source means business. It's only when people misunderstand how software is paid for that anyone would think otherwise.

The key insignt that many people miss in considering the economics of Open Source is that almost all companies are software users, and many companies contract or develop custom software for their own use, yet few companies are in the business of selling software. Most companies use software to facilitate some other sort of business – selling books, for example. For the bookseller, most computer software is a non-differenatiating cost-center: a necessary piece of infrastructure that makes his business neither any better nor any worse than his competition. The competition has to solve the same problems.

For example, when I worked at Pixar, our business was making movies, but we used, and wrote, software for movie-making . You can find my name in the credits of "Toy Story" and "A Bug's Life", it's on the screen right after the out-takes end.

To make those films we made a lot of software. Some of it made a difference to Pixar's product, such as our animation and rendering software. That was software that the company kept for itself, or sold for a high price, and rightly so. That's the sort of software that should remain proprietary, because it differentiates the company that creates it. In Pixar's case, that software made films possible, and contributed to the unique look of the films.

And some of our software at Pixar helped us do our job, but didn't differentiate the company, such as a memory allocation debugger I wrote there called "Electric Fence". That software would take a problem that had been very difficult to debug, and find it instantly. So, in 1989, I released the Electric Fence source code on Usenet, as Open Source. I hadn't written full documentation for it, since I was lazy and it wasn't part of my job anyway. But the next day, a total stranger emailed me complete documentation for Electric Fence. That person wanted to deploy my program in his company, but needed documentation to do it. And now I could use that documentation at Pixar, too, and more people would be able to understand my program and use it effectively there. So, sharing my source code had gotten the company some useful documentation for nothing.

In a similar way, programmers at the companies and organizations that needed to use the web created the Apache web server. Each collaborator had a purpose to achieve for his employer or his own business, whether it was to put homework assignments on the net or to sell books. Many of those people were on salary, but few of them had software sales as their primary business. Together, they made the server that dominates the web.

So, companies pay for the software they need, whether it's Open Source or proprietary. When they pay for producing Open Source, they share the cost of development with other companies that are interested in the same project. Thus, nobody pays too much or risks too much.

Wouldn't it work just as well for them to buy proprietary software rather than to develop Open Source? It turns out that proprietary software often loses on three points: control, visibility, and cost-effectiveness.

Regarding control: imagine having to convince a large software company to put in the feature that you need. Do you have to convince their marketing department that enough customers will want it? In general, you have much more control over Open Source, because you can have modifications made at your own expense and then have them accepted into the main code tree. You'll often find that a company like yours has already contributed some of what you'll need.

Regarding visibility: in 1995, Microsoft told the Hewlett Packard Company that Microsoft would soon have an enterprise-quality NT system that would kill Unix in the market. In response, a vice president at HP de-emphasized development of HP's own Unix system. Of course, Microsoft slipped the release of an enterprise-quality NT by 8 years, if they can be said to have one now. HP owned most of the scientific workstation market until that decision in 1995. Their decision to believe Microsoft's NT release date and let their Unix systems lag behind other brands gave their workstation market to Sun Microsystems, and was the root cause of HP's recent financial problems. HP's problem was that they didn't have visibility of the day to day development of NT, and couldn't make an accurate asessment of its prospects.

In contrast, companies have much more visibility to Free Software development. Most Open Source projects have their source-code control system available for anonymous access.

You can get an up-to-the-minute asessment of development, and make your own forecasts. If your company feels that some aspect of an Open Source project is lagging, you can get your own developers on to the project, to address the specific issue you care about. But you won't have to shoulder the burden of the entire development.

And the third issue is cost-effectiveness. Consider how your software dollar is spent when it comes to retail software: some of it is spent for advertising, package design, to pay for space on retail dealer's shelves, and for many other things that are critical to the retail software vendor's business but aren't software development. So, perhaps ten Yen out of every hundred actually pay for software development. In contrast, when you pay for development of Open Source, most of your money goes into software development.

So, what do you do with the money you save? That money doesn't disappear from the economy. You put it into something that's important to you. Maybe you put it into advertising your own business, or designing packages for your own product, or you just add it to your bottom line. Either way, you pay the same taxes, you support the economy and education, but one way gets you more out of what you spend on software than the other.

So, should we make Open Source cars, then, and eat Open Source food? Not until somebody invents the "Star Trek" replicator, a copying machine for physical objects. That's the critical difference between software and other products. It takes land, water, sunlight, and labor to copy a bowl of rice. But it takes much less to copy a software program. So, in economic terms, Open Source is a "non-rivalrous public good" [translator: "rivalrous" - being a rival]. If one party uses it, that doesn't hurt any of the others using it. Generally, any nonrivalrous public good is a positive contribution to the economy, except to the individual who happens to have made a private good, like proprietary software, that suddenly has competition from a public one. And there are other products that have similar economics to software: for example, there is now a large Open Source community for VLSI hardware designs. [opencores.org]

So, here is a tremendous change, called Open Source, that is the product of individuals and companies coming together in a collaborative and democratic way on the internet. And it's hardly a surprise that the internet depends on the very software that it helped create. The domain name system, the Apache web server, and the operating systems underlying the internet are Open Source.

But there are clouds on the horizon. There's already something close to a monopoly regarding the web browser: many sites report that 95% of their viewers are using Microsoft Internet Explorer. There's also a telecommunications monopoly, as many governments grant a monopoly on the internet connection, to homes and businesses to a single company within a huge geographical area. This was not as much of a concern with telephone companies as it is with the internet connection, because telephone companies did not control the content of your call. There's a technological challenge, called "Palladium" or "Trusted Computing", which threatens to take control of their computer away from the computer user. And there are legislative challenges such as software patenting, which is made even worse when patented technologies are embedded in industry standards, denying interoperability to anyone who can not get a patent license.

These factors are challenges to the democracy of the internet, and also challenges to Open Source software.

Let's start with the Internet Explorer issue. Many commercial sites now develop exclusively for Internet Explorer. What they should be doing instead is development to industry standards. Users of other browsers than IE are denied access to banking, shopping, business, and even web communication with their own government. One way that government can help is by insisting that government sites be written to accomodate open web standards rather than a single browser.

Regarding telecommunications: we now have a "duopoly" [translator: "duopoly": monopoly of two rather than one] where I live in California. You can get internet service from the phone company, or the cable TV company, although the cable service isn't appropriate for business. Previously, multiple companies were allowed to use the telephone wires, but now we are down to one. That company can deny access to the internet, and has content restrictions as part of its customer agreement. One way to help with this situation would be to encourage the development of wireless "mesh" networks, using spread-spectrum radio, to provide homes with Internet access. It's possible for these networks to operate cooperatively, with small vendors providing local wired internet connections to a neighborhood and without a single company controlling the net.

The technological challenge might be the most serious one, if users accept it. It's been called "Palladium" or "trusted systems". In "trusted systems", the party that isn't trusted is the owner or user of the computer. Those systems have hardware and software that is meant to constrain the user's actions.

Computers incorporating trusted systems will be built to prevent their user from modifying software. Web servers incorporating trusted systems will only accept connections from software that has been certified by some sort of authority - and if you modify one byte of that software, you lose the certification automaticaly. A cryptographic mechanism is used to enforce this. Only those who are certified are given the cryptographic keys required to access the web, or even to run a program on a computer. The computer hardware is made to check for modifications and to remove privileges when they occur.

Palladium and trusted systems are obviously extremely hostile to Open Source software - so hostile that they could have been designed to cut us out. Open Source depends on people's being able to modify software, and participate in network commerce using that software. In response to this, the creators of "trusted systems" say that anyone who doesn't like them can simply turn the trusted feature off. What they don't mention is that sites that have the trusted feature turned on won't communicate with ones that have turned it off. So, Palladium is designed to make an uncommunicating island out of Open Source, and out of any software that isn't certified.

There are some attractive and beneficial features to Trusted Systems and Palladium. They may enable forms of web commerce and data security that we don't have today. But the bad effects are bad enough that we must be extremely careful in deploying this technology.

DRM - Digital Rights management, is a similar technology, and also presents us with problems. Although we have digital video disk player software in Open Source, that software is not legal in many nations. This is a disappointment to those of us who only want to use the software to play DVDs on our Linux laptops. The companies that have made the copy protection for DVD discs will not grant us a license to implement it, because they are afraid that an Open Source player program would be modified to make unprotected copies of the DVDs. DMCA - the Digital Millennium Copyright Act, is a law in the United States that makes our software illegal. This law goes too far - rather than penalize legitimate uses of software, such as playing a disc that I've bought on my own computer, enforcement should be directed only at the people who make illegal copies. The problem with DMCA and DRM for us is that it shuts Open Source software out of media access, and thus makes Open Source less desirable to use. If we get too many applications for which it isn't legal to use Open Source, nobody will be able to use it for anything.

Software patents are also a tremendous threat. Since Open Source developers generally do not collect revenue from software sales, they aren't bulding up a budget to defend themselves in court. If the very act of producing software places significant legal liability upon them, they aren't going to be able to produce Open Source.

This is a serious problem in areas where software patenting is legal, since it is possible for the patent holder to make a frivolous or unsubstantiated claim of patent infringement. It can cost you up to one Million US Dollars to defend yourself from a patent infringement claim. Most Open Source developers can not afford a single day in court, and would be required to settle with their opponent, regardless of the merits of the case. The chance of a frivolous claim are very high, because software patents generally are written to be deliberately vague so that they can be applied as widely as possible. In the US, many software patents are granted for things that have been invented before, by other people. Patent examiners are not granted adequate time to examine prior art. In the US, they often have only two or three hours to examine an application, and are rewarded according to the number of patents they approve rather than reject. So, we end up with very many low-quality patents in the US, and we determine if they are valid or not in the courts rather than the examiner's office.

The situation of patents is made worse when standards organizations incorporate patented technology into industry standards. Once that happens, anyone who can not get a patent license is now allowed to interoperate with other systems using that standard. To respond to that, some standards organizations have settled on something called RAND, for Reasonable and Non-Discriminatory patent licensing. But they don't have a good definition of "Reasonable". Every web click rests on thirty seperate standards, from the cable connectors through the software protocols. If each of those standards contains a single patent, and each patent requires a 3% royalty, small and medium-sized businesses will be shut out of implementing that standard. Only very large companies, which have patent cross-licenses with each other and thus don't pay each other royalties, will be able to implement the standard.

And Open Source won't be able to implement standards with RAND licensing at all. Since we don't charge royalties for our own software, we can't pay patent royalties. To be impelentible by Open Source developers, any patents incorporated into industry standards must include a royalty-free license for the purpose of implementing the standard. Patent holders would still be able to get royalties for any other use of the patent than an implementation of the standard.

So, those are the challenges we face. If we are able to manage those four possibilities, we may be able to preserve and inprove the democracy of the Internet, and make the world a safe place for Open Source.

Now, I'd like to talk about a problem that corporate and government purchasers face today, called the Addiction Model of Purchasing. This is the way that Microsoft locks in entire markets. In the addiction model, a product is originally introduced at low or zero cost - for example shipped as a free accessory with the operating system. Microsoft can afford to do this, because they figure they'll make their money in a few years, when you have to upgrade the software. The addiction part comes because the product is deliberately incompatible with other similar products, and each upgrade makes a change that is incompatible with all older versions. Here's an example that many offices have faced. One office upgrades its version of Microsoft Word, and then if the other offices want to be able to read the files created in that office, they have to upgrade their copies of Word as well. Microsoft has effectively shouldered their way through the market [translator: the metaphor is one of an impolite person pushing his way through a crowd] this way.

Addiction Model vendors use incompatible file formats and incompatible intercommunication protocols, rather than standard ones, so that you are compelled to purchase their own brand of client to work with their servers. For example, if you get the Outlook client, you're going to have to install an Exchange mail server.   If you want to change your server from Exchange to something open, you'll have to replace the Outlook clients on every desktop. If you have 80,000 desktops, changing one server may mean changing all of those desktops, and it's too expensive to contemplate.

So, the result of purchasing from an Addiction Model vendor is that your entire shop eventually runs that vendor's software, because any other way of operating is too difficult. And since that vendor has you addicted, they can charge what they want, and their service and software quality can be as bad as they like. It would be so expensive for you to switch brands that you'll stick with them. Of course the effect of the artificially high prices is that corporate profits are lower, and taxes are higher because government users are spending too much on their software.

So, now that you're addicted, how can you break the habit? You can do so with your own purchasing policies. First, you must recognize in your policies that larger long-term savings are more important than smaller short-term ones. Otherwise, you'll be tempted to take that deep Microsoft discount and get hooked. Once you understand that, you can implement simple policies that cause a long-term cost reduction:

Require that the file formats and intercommunication protocols of the products that you purchase be publicly documented, and available for all competitors to implement without a royalty or discriminatory license. They don't have to be formal standards, they just need to be disclosed and have rights available. So, you'll have servers and clients from different manufacturers, that interoperate with each other. You won't have to replace 80,000 clients when you replace a server. And you'll have a competitive market, with the lower prices and higher quality that come when capitalism works the way it should.

But how will your vendors keep making money if you don't allow them to differentiate themselves? They shouldn't have a problem, because so little of what makes their software special is in the file formats and intercommunication protocols. The stuff that makes their software special is in the design, the logic of the program, and the user interfaces. Remember, we aren't proposing that these vendors go Open Source. We're just proposing to require open file formats and intercommunication. However, the folks who can't compete fairly, on the merit of their own software, are going to have a problem with that - because they're losing the right to lock you in.

So, imagine that we do achieve a fair market for software, in which Open Source and proprietary software can compete fairly upon their merit. Suppose that a software purchaser is faced with a choice between two functionally equivalent programs, one Open Source, and the other proprietary. Both programs do the same things, and each works as well as the other. Are the two programs equal in merit? Not at all. Consider:

* One product can be freely redistributed, the other requires a payment for each user.

* One can be modified as you like, the other only if you convince the vendor.

* One offers a choice of service vendors, the other has a service monopoly.

* One would survive the demise of its vendor, the other might become un-maintainable.

* One can be audited for security issues, the other is opaque.

Are the two products really equivalent in merit? Of course not. So, you can see that when comparing two programs, intellectual property policy is a merit to be considered, along with functionality and quality.