Free (Libre) and Open Source Software: A Social Justice Primer for Churches

You will be made rich in every way so that you can be generous on every occasion, and through us your generosity will result in thanksgiving to God. 2 Corinthians 9:12 New International Version (NIV)

We do not need to artificially restrict the infinite supply of digital goods, in order to ensure the creators and providers of those goods are properly rewarded (potlatch.net, 2001)

What is F(L)OSS? Why should we care?

Faith-based organizations around the world have been leaders in drawing attention to a variety of social justice issues, from globalization of trade, to ecological devastation and self-determination rights of indigenous populations. However there is one key social justice concern that has escaped the notice of virtually all religious organizations: Free (Libre) and Open Source Software or F(L)OSS. This failure is becoming more and more important, as religious organizations seek to extend their reach on the Internet to a younger, more technically aware generation. This generation has a finely tuned sense of irony, and can sniff out apparent hypocrisy from a mile away. Although churches have already discovered the preference of individuals aged 30-45 for electronically mediated two-way communication, they have so far largely failed to understand how their institutional choices of proprietary technology present an apparent contradiction in the eyes of their high-priority youth target audience. This failure opens religious organizations to criticism, or worse, to indifference.

In this background paper, we will examine some of the tensions underlying the notion of F(L)OSS, enabling individuals and religious organizations concerned with social justice to determine their responses. We will highlight some of the parallels between F(L)OSS communities and religious organizations, in particular, their organization, as **gift economies**.

F(L)OSS? Open Source? Free Software? Help – I'm confused!

When we use the term "Free" in conjunction with software, it has the meaning understood in French as "libre" – referring to liberty rather than price. To make this clear, the letter "L" often appears in the acronym. Because access to the "source code" (instructions to computers that are readable by humans) is considered essential to ensure the liberty of software, some people prefer the term "Open Source Software", or OSS. We will use the combined acronym "F(L)OSS" throughout.

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

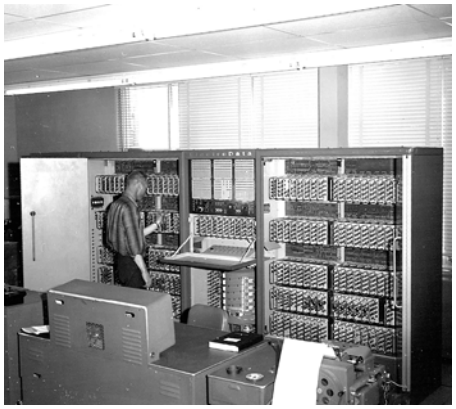
- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3).

(Free Software Foundation, 2005)

Unfortunately there is another similar term, "Freeware", which refers to software that is made available at no monetary cost to the user, but under restrictive licensing terms which disqualify it from the definition of "Free Software". Good examples of Freeware are the Internet Explorer web browser from Microsoft, and the Acrobat Reader program from Adobe Systems. These programs cost nothing to download, but are not "Free Software" or F(L)OSS.

A Historical Perspective: The Rise of Software-as-Product

Today we take for granted that everyone understands the term "software" as "the entire set of programs, procedures, and related documentation associated with a system and especially a computer system" (Merriam-Webster Online Dictionary, n.d.). But the idea that such programs and procedures could have value outside their original social contexts is a relatively recent one. For many years, computers were exceedingly expensive, and therefore purchased only by large institutions who hired programmers to write programs that made them useful in a particular social context (Raymond, 1999). Although the term "software" was not widely used in the 1960s and 70s, it would have been understood then as referring to a service provided by computer programmers.



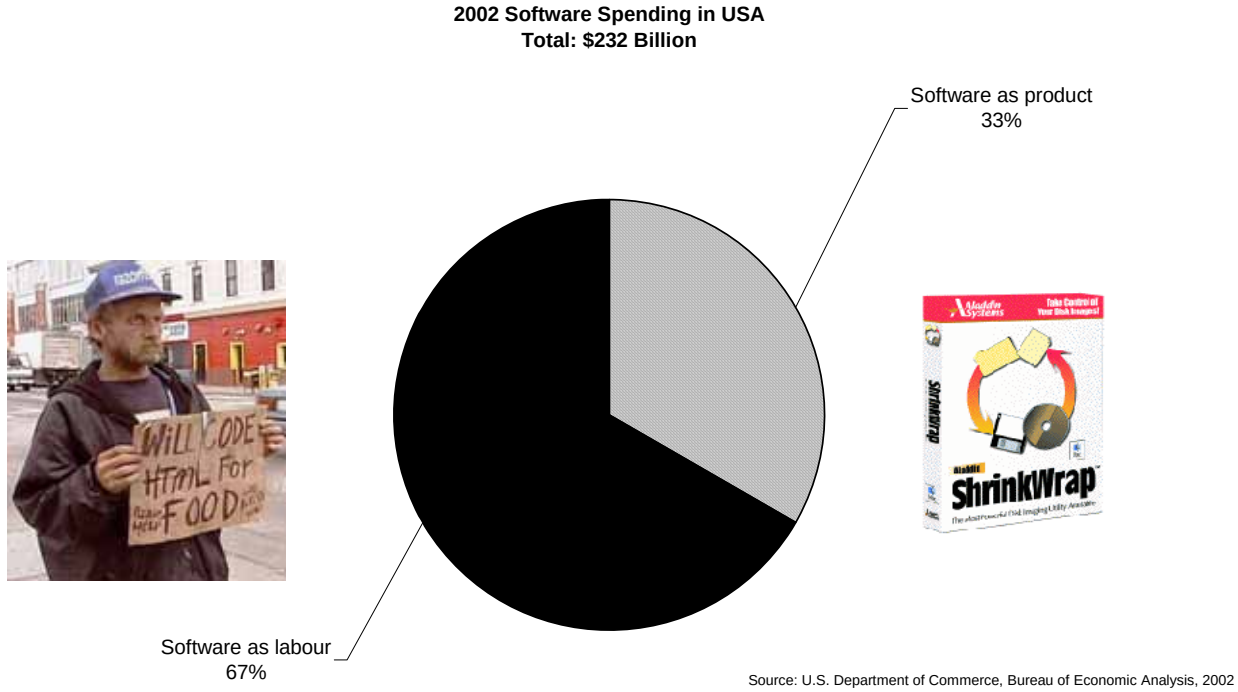
An early mainframe computer
Source: US Geological Survey, 1964.

However, with the advent of the personal computer in the late 1970s and early 1980s, this changed completely. Once stand-alone computers were available cheaply to a mass market, it became clear that supplying programs for these computers represented a substantial opportunity for the concentration of wealth. The way to make money from software development was to identify a simple need shared by a large number of users and write a computer program to meet this need. Because at the time few computers were connected to any form of network, the most efficient way to distribute such programs was to embed them in portable media such as floppy diskettes, place them in a shrink-wrapped box with a printed user manual, and ship them to stores where people could buy them. This distribution method reinforced the notion, now widespread, of software-as-product.

As soon as the competitive market of software-as-product became established, suppliers identified that minimizing interoperability with competitors' programs was a key strategy for achieving market dominance. In a market where all suppliers are pursuing this strategy, the risk is high that the market will fail into a monopoly: a "winner take all" scenario. Such a market failure brings to society not only costs but also at least one significant benefit: the creation of *de facto* standards through the effective dominance of one supplier's products or services. The costs to society include, of course, the excessive concentration of wealth and power in the hands of the monopolist organization and its shareholders. This theoretical failure of the global software market is exactly what actually occurred during the 1990s. The complete dominance of Microsoft Corporation led both to the creation of standards that were otherwise lacking, and to the extreme concentration of wealth and power in the hands of an unelected few.

Elected officials in numerous jurisdictions around the globe responded to this situation through a variety of political and legal means, with not much success. Microsoft to this day remains so wealthy that it can essentially afford to give away at no cost almost any software that a competitor might create. This anti-competitive practice was the subject of a lawsuit by a consortium of US states filed in 1998. Microsoft eventually consented to a judgement in 2002 which acknowledged its unlawful behavior in exchange for minimal penalties that were largely irrelevant by the time they were imposed (*State of New York et. al. v. Microsoft Corporation*, 2002).

The emergence of the market for software-as-product and its subsequent failure into monopoly were contingent on two factors: 1) the ability of software producers to exclude competitors from the source code of their applications; and 2) the protection of the legal schemes of copyright and patent that are imposed politically on the software



Source: U.S. Department of Commerce, Bureau of Economic Analysis, 2002

market. Both of these factors need to be further explained in order to understand the problem that F(L)OSS solves.

Every piece of software is written by developers in a human-readable computer language, which makes up the source code. This code is then converted into machine language, which is what you purchase on a disc in a box from a store. The source code is never included when you purchase software-as-product. (In fact, if you take the time to read the tedious legal language of the end-user license agreement or "EULA", you'll probably find that in many cases, you are not really acquiring much at all in the way of property when you pay your money to the merchant and carry home your shrink-wrapped box). Since the very act of using software creates in the user the need to change it (Peizer, 2003), this presents a bit of a problem. Of course the market for software-as-product might offer you a solution if your needs are typical enough. Likely there's some add-on, or companion product that modifies your original purchase in some small way (but which cannot

What is Source Code?

A computer is a machine which essentially consists of a lot of on/off switches. In order to make these switches do something useful, we have to plan out which switches to turn on in what order. This is where the ones and zeros (bits) come in. We call these "machine code" because they tell the machine which switches to turn on or off. In very early and simple computers, the programmers would actually write their machine code by hand on paper and then toggle it into the switches on the front. As software got more complicated, this machine code got longer and longer. Today's software is constructed of billions of bits! To make really complicated software, it is much more effective to write what we want to do in a high-level language (one that looks something like English) and then have a program (a compiler) translate this into machine code.

Source Code	Machine Code (bits)

`#include <stdio.h>`
`main () {`
`  printf("hello world\n");`
`}`
	`11111110010100010000`
	`01000110001001000001`
	`00000000100000000000`
	`00000000000000000000`
	`01000000000000000011(this goes on for thousands of lines)`

itself be lawfully modified by you in any way) that makes it more useful to you.

This very tension is what drove the foundation of the F(L)OSS movement. In the early 1980s, a computer scientist named Richard Stallman was working in the Artificial Intelligence Lab at MIT. Stallman had been having problems with a printer, and wanted to modify the source code of the printer control program to make it work properly. But the individual who wrote the program "refused to give [Stallman] and the MIT AI Lab the source code for the control program for [the] printer" (Stallman, 1999) because he had signed a nondisclosure agreement a with a producer of proprietary software and hardware.

Reacting to this experience, Stallman issued a manifesto. In it, he defined four important freedoms that would define free software: 1) users have the ability to use the software for any purpose, 2) the software is open to being improved to meet the needs of the user, 3) users are able to distribute the software to anyone who finds it useful, and 4) those who make changes and/or improvements on the software are able to distribute the changes/improvements they made. It is important to realize that here 'free' does not refer to the idea of 'no cost', but rather to the notion of liberty. Users may have to pay a price for the software, but they are free to do what they please with that piece of software. Because the word 'free' in English language is ambiguous, we often use the French word 'libre' to define free software, hence the term 'F(L)OSS': Free (Libre) and Open Source Software.

But it is not only by withholding the source code for their programs that the makers of software-as-product were able to exclude purchasers from acquiring the rights that Stallman envisioned. Software is itself a type of information, and therefore meets economists' definition of a "public good". This means two things: 1) my use of a piece of software does not lessen your ability to use it simultaneously; and 2) with the exception of source code, it is almost impossible for me to keep you from copying it (Hawkins, 2004). Economic theory would predict that because software is a public good, markets will fail to encourage the production of enough of it without some form of political intervention. The result would be unmet demand for software. As a result, Western society has devised various intellectual property protection schemes to encourage innovation and discourage undersupply (Stiglitz,1999). However, it's important to realize that these intellectual property laws and treaties were implemented many years ago, in a time of information *scarcity*. In the context in which they were devised, they balanced the need to give creators adequate incentive to create new information against the public's need for unrestricted access. Many authors have recently commented on how current intellectual property laws and treaties no longer balance these needs effectively, because we have entered an era where information is globally *overabundant* (e.g. Lessig, 2000). However, in the early 1980s, in none of this more recent legal and economic scholarship existed.

What Richard Stallman devised was a way to use the existing intellectual property law of the day to ensure and promote software freedom, rather than to restrain it. He did this through the creation of a legal document called the General Public License (GPL). This document, when used by the creator of a piece

Richard Stallman, 1983 from: www.stallman.org

of software, allows that person to retain copyright control of his or her work, and yet make that work available to others in a way that embodies Stallman's four freedoms. The GPL itself contains within it provisions which allow it to be distributed under the terms it describes, but which allow its control to be retained by the Free Software Foundation, an organization established by Stallman expressly for this purpose.

Subsequently, others have found the terms of the GPL too restrictive for their own purposes, and have created similar licenses that reserve different sets of rights to their originators. Among the important variants is the Berkeley System Distribution (BSD) license. Although the GPL does allow derivative works to be distributed commercially (i.e. sold), it states that they must *also* be made available free of charge. Furthermore it restricts incorporation of any part of GPL-licensed work into a product that does is not also distributed under the GPL. The BSD license does not contain these restrictions (Krishnamurthy, 2003). An example of the importance of this distinction is the Apple OS-X operating system. In creating OS-X, Apple used substantial portions of FreeBSD, a F(L)OSS operating system, but neither made OS-X available for free, nor released the source code for it. Apple would not have been able to carry out this action if it had based OS-X on Linux, a different F/LOSS operating system that is licensed under the GPL.

Stallman not only released the GPL, but wrote a substantial number of important computer programs and released them under the GPL. However, his work didn't get much attention until a couple of other important things happened. The first of these was the advent of the Linux operating system.

In 1991, Linus Torvalds was studying computer science as an undergraduate in Finland. Because of his academic affiliation, he had relatively easy access to Stallman's Free Software, which was written entirely to run on the large, shared computers found at universities and other institutions. Torvalds thought that it would be "cool" to try to adapt Stallman's programs to run on the cheap, personal computer hardware of the day. To do this, he needed to create the "kernel" of a new operating system for personal computers: the most essential, lowest level program that communicates with disks, video displays, keyboards, etc. Unlike Stallman, who is described as both a loner and a genius, Torvalds preferred to work in groups. Using the Internet (but not the worldwide web, which wasn't invented yet) he invited academic computer scientists around the world to join him in creating this new operating system, which he called Linux. This self-organizing network of volunteers soon started to break new ground not only in the field of computer science, but also in the way that they worked collaboratively together on a project from which none of them would derive significant, direct monetary benefit. This phenomenon of massive, distributed, self-organizing volunteer labour continues to accelerate to this day (Raymond, 1999). The exact nature of the motivation for participation in such projects is the subject of much recent academic work. (Haruvy, Prasad and Sethi, 2005; Benkler, 2001). Nevertheless, the end result is widely accepted as being a "gift economy" – where goods and services are exchanged without direct *quid pro quo*, and where a participant's power and status are derived not from what s/he has accumulated by *taking* from others, but from what s/he has contributed by *giving* to others (Pinchot, 1995).

Although the F(L)OSS gift economy has existed since the advent of computing in the mid-twentieth century, it is only recently that advances in computer hardware and networking technology have made it feasible for virtually every person on earth to participate in it. Yet we stand at a point in history where every individual and every organization in all developed nations (and most developing nations) are faced with important choices about their level of participation in the F(L)OSS community. In the next section, we will examine the ramifications of these choices.

Implications for Churches

Scarcity or Abundance?

Few would debate that we live in an age of information abundance. Some would even argue that, at least globally, the abundance we experience extends beyond the realm of information. The problem most of us perceive is not how to get access to sufficient amounts of information, but rather how to sort through the mountains of information directed at us and determine which, if any, is useful to us in our personal and social contexts. On the other hand, the system of intellectual property law under which we currently operate embodies the assumptions of information scarcity that were generally true in the late eighteenth century. The F(L)OSS phenomenon has demonstrated that at least in some cases, political intervention in the market for information is not required to ensure adequate production of information. Instead, such protection leads to both market distortion (excessive wealth accumulation) and eventual market failure (monopoly). In the past, churches have typically been quite fearful of breaking the law, and so have adhered quite strictly to existing copyright law. In addition, many church musicians and composers feel a sense of comfort with the status quo, and hesitate to upset the cart on which they've been riding, even if it is small and slow-moving. However, the time may soon be at hand for churches to consider whether support of the existing copyright regime is consistent with their notions of social justice.

Failed Market or Gift Economy?

In deciding not to challenge existing intellectual property law, churches are not only preserving a political scheme for ensuring adequacy of innovation that may no longer be necessary. They are also rejecting an important alternative that has substantial consistency with the theological and social principles on which churches operate: the notion of gift economy.

It's fairly easy to see how individual Christian congregations operate as gift economies, since the ideas of tithes and offerings are central to their fiscal foundations. At least within the Christian context, however, it is possible to find homology with gift economies at an even deeper level. It does not seem like too much of a stretch to say that the very foundations of Christianity can be thought of as the transition from a *quid pro quo* system for exchanging love (or spiritual favour) whose transactional rules were enforced contractually through the ten commandments, to one of unrestricted abundance of love and grace through the ultimate sacrifice of Jesus Christ. This argument seems worthy of careful consideration and reflection on the part of individual Christians and Christian churches. Canadian churches should find this argument particularly poignant, since the indigenous peoples of Canada are often cited as having well documented gift economies prior to their being assimilated into the global capitalist market.

Consumers or Citizens?

The advent of the Internet was supposed to reduce some of the power asymmetries inherent in the control and ownership of media by making it economically feasible for every citizen to be a content creator and publisher (Schement and Curtis, 1995). This promise, however, has largely not been realized. In fact, in his article, The Real Digital Divide: Citizens versus Consumers (2002), Oscar Gandy looks at how new forms of media have actually widened the gap between citizens and consumers. The content of the Internet is still produced mostly by professionals, and consumed passively by individuals supported by advertising. Now, however, the clicking habits of users on Internet sites is simply added to the large body of other data (such as credit card purchases) that enables the classification of individuals into groups which serve the needs of a capitalist market. This opportunity has not been lost on the few remaining software monopolists in each sector. Foreseeing the demise of software-as-product, both Apple and Microsoft have aggressively allied themselves with media concerns (Apple with Disney/ABC, and Microsoft with NBC). The net result is that the view of individuals-as-citizens on the Internet is discounted, while the view of individuals-as-consumers is validated and supported. The point of all this is that when churches continue to support monopoly-seeking makers of proprietary software,

they also appear to validate the notion of individuals-as-consumers, which they generally otherwise seek to diminish.

This apparently contradictory behavior is not lost on many young, computer-literate would-be church-goers. Because of their overall low level of technical sophistication, Church organizations probably aren't aware that their web visitors can easily identify the technology that an organization is using to serve its web pages. If a church, which is trying to encourage young people to participate in a gift economy, uses proprietary (e.g. Microsoft) technology to serve web pages, or to facilitate on-line communities, many socially and technically conscious young people simply cannot disregard it. Although there is no literature to suggest this, it seems like a plausible reason why churches might have trouble attracting young and tech-savvy new members. If this is true, it represents an important lost opportunity for churches. Potential members who might be repelled by seeing proprietary technology on a church website are disproportionately valuable as potential church members, since they're often already well-versed in how to administer an effective gift economy, having probably participated in several already.

Self-determination or Dependence?

Because of the near-zero acquisition cost of F(L)OSS, many organizations that apply information and communication technologies in the developing countries of the world use F(L)OSS extensively. Using F(L)OSS reduces the costs to these organizations substantially, and allows workers in developing countries to acquire computer skills while adapting software to their own unique social contexts. In addition, "many developing countries are reluctant to uphold intellectual property laws or agreements that make access to information more costly, impede technology transfer and increase the monopoly power of multinational corporations" (Stein & Sinha, 2002, p. 412). The fact that makers of proprietary software and their billionaire founders are often quite generous in donating resources to the poor in no way reduces the inequity of the wealth accumulation in the first place. Furthermore, the fundamental issue of self-determination remains. Why should poor people pay money to Microsoft for software which they largely don't need, so that they can receive some of their own wealth back with strings attached?

Summary & Conclusions

In conclusion, one can see that F(L)OSS essentially embodies the notion of software-as-service, and creates a tension with the more common, but relatively recent, view of software-as-product. Software-as-product implies a conceptualization of information as being disconnected from social context, and requires political intervention to maintain adequate levels of innovation. When supported in this way, however, software-as-product has a natural tendency to lead to market failure by monopoly. On the other hand F(L)OSS, or software-as-service, relies on the notion that like all information, software requires extensive social contextualization to be maximally useful. Although innovation in a F(L)OSS market might proceed more slowly at first than in a market supported politically through restrictive intellectual property rights, the decentralized production and consumption of F(L)OSS eventually leads to a gift economy where software is priced at its true marginal cost: an ideal market (Klemens, 2006, p. 97). Stakeholders win or lose in the tension created by F(L)OSS according to their degree of alignment with the centralized, monopoly-seeking world of software-as-product, or the decentralized gift economy of software-as-service.

The implications for churches are many, and important to consider. Churches face important choices today about whether they will continue to lag behind other organizations who have recognized the importance of F(L)OSS and begun to support it actively.

Bibliography

Benkler, Y. (2001). The battle over the institutional ecosystem in the digital environment. *Communications of the ACM*, 44(2): 84-90.

DeLong, J.B. and Froomkin, A.M. (2000). Speculative Microeconomics for Tomorrow's Economy. *First Monday*, 5(2), [online]. Retrieved August 23, 2006 from http://firstmonday.org/issues/issue5_2/delong/index.html

Free Software Foundation. (2005). The free software definition. Retrieved November 23, 2006 from http://www.fsf.org/licensing/essays/free-sw.html.

Gandy Jr, O. (2002). The Real Digital Divide: Citizens versus Consumers. *Handbook of New Media*. (pp.448-460). London, UK: Sage.

Goth G (2005). Open Source Business Models: Ready for Prime Time. *IEEE Software*. 22(6), 98-100.

Haruvy E, Prasad A, Sethi SP. (2003). Harvesting Altruism in Open-Source Software Development. *Journal of Optimization Theory and Applications*. 118(2), 381-416.

Hawkins, R. E. (2004). The economics of open source software for a competitive firm. *NETNOMICS: Economic Research and Electronic Networking, 6*(2), 103-117.

Klemens, B. (2006). *Math you can't use: Patents, copyright, and software*. Washington, D.C.: Brookings Institution Press.

Krishnamurthy, S. (2003). A managerial overview of open source software. *Business Horizons*, 46(5), 47-56.

Lerner, J. & Tirole, J. (2002). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50(2), 197-234.

Lessig, L. (2000). Open code and open societies. In J. Feller, B. Fitzgerald, S. A. Hissam, & K. R. Lakhani (Eds.), *Perspectives on free and open source software* (pp. 249-360). Cambridge, MA: MIT Press.

Merriam-Webster Online Dictionary (n.d.). Retrieved November 25, 2006, from http://www.webster.com/dictionary/software.

Peizer, J. (2003). Realizing the Promise of Open Source in the Non-Profit Sector. Retrieved December 3, 2006, from www.soros.org/initiatives/information/articles_publications/articles/realizing_20030903

Pinchot, G. (1995). The Gift Economy. Retrieved November 26, 2006 from http://www.context.org/ICLIB/IC41/PinchotG.htm.

Raymond, E. S. (1999). *The cathedral & the bazaar: Musings on linux and open source by an accidental revolutionary*. Sebastopol, CA: O'Reilly.

Schement, J. R. and Curtis, T. (1995). Chapter 1 – The New Industrial Society. In Tendencies and Tensions of the Information Age: The Production and Distribution of Information in the United States, (pp. 21-45). New Brunswick, NJ: Transaction Publishers.

Stallman, R. (1999). The GNU operating system and the free software movement. In C. DiBona, S. Ockman, & M. Stone (Eds.), *Open sources: Voices from the open source revolution* (pp. 53-70). Sebastopol, CA: O'Reilly & Associates.

State of New York, et al. v. Microsoft Corporation, No. 98-1233. United States District Court for the District of Columbia. November 1, 2002.

Stein, L. & Sinha, N. (2002). New global media and communication policy: The role of the state in the twenty-first century. In Lievrouw, L. & Livingstone, S. (eds.), *Handbook of new media*, (pp. 410-431). London, UK: Sage.

Stiglitz, J. (1999). Knowledge as a global public good. In I. Kaul, I. Grunber, & M. A. Stern (Eds.), *Global Public Goods: International Cooperation in the 21st Century* (pp. 308-325). New York and Oxford: OUP.

Stone, A. (2002). Why businesses use OSS. *IEEE Software*. 19(2), 102.

US Geological Survey. (1964). Retrieved December 11, 2006, from http://libraryphoto.cr.usgs.gov/htmllib/btch545/btch545j/btch545z/pap0018b.jpg

Wilhelm, E. L. (2005). What is source code?. Retrieved November 26, 2006 from http://scratchcomputing.com/articles/whatis_source.html.