Alexis O'Connor
Kian Win Ong
Ted Sander
Matt Ferlo

Government Policies on Open Source

Introduction

Open source software is a relatively recent phenomenon that has been steadily gaining in popularity. Rather than hiding their source code away from the world, practitioners of open source allow everyone to see it. The majority of them even allow anyone to copy it and make changes to it. Through this revolutionary model, many impressive pieces of software have been made – most prominently, Linux. One could argue that the Internet is run by open source software, in the form of Apache, Sendmail, PHP, Perl, etc. The open source concept is a powerful new paradigm that's here to stay.

Given that fact, should the government do anything about it? And if so, what? Those are the questions that we attempt to examine in this policy brief. While many corporations have taken definite stands (both for and against) on open source, many governments have only just begun to do so. There are many reasons for the adoption of open source in governments, and often there are as downsides. We will attempt to look at these, and also analyze what actions have been taken and policies implemented by various governments.

Governments are in a unique position in almost any industry. By the nature of their immense size and influential position, their actions may have far-reaching effects. In the field of software, the programs used by a governmental agency often have a distinct advantage compared to unused programs. Software that is used becomes the *de facto* standard when dealing with that agency – if, indeed, it is not mandated. In addition to its own usage, the thousands of government contractors are also forced to adopt the government's software platform of choice, so they are eligible to work for them.

When all of these factors are considered, it becomes clear that the government plays a major role in determining the fate of the software industry, *whether they intend to or not*. Even the lack of an official policy is, in a sense, an action. Given this, governments should take time to consider the various factors in the open source/closed source debate, and choose an appropriate policy. Even if it is decided that there action should be to maintain the *status quo* and issue no official announcement, it is better to do that having considered the options than to do so out of ignorance or laziness.

We believe that there are fundamentally three broad actions that the government could take with respect to open source. They could promote open source solutions over closed source solutions (even going so far as mandating open source). They could promote closed source solutions (similarly, even so far as banning open source). Or, they could consider the two approximately equally, and choose the best software for the task at hand.

We will consider those three positions in some detail. First, we will examine the benefits, drawbacks, and other considerations of a policy explicitly promoting open source. Then, we will look why it may be beneficial to ban or discourage open source

use, as well as some of the ramifications that this may present. Finally, we will consider a series of questions that a government should ask itself when choosing between open and closed source.

Government Promotion of Open Source

The phenomenon of governments formulating legislation and policies on open source software is a recent trend that seems to be gathering some momentum in certain regions. Considering the lucrativeness of government contracts, governments awarding large contracts to open source solutions have been prominently featured in the business press. The city of Munich in Germany is spending 30 million Euros to convert 14,000 desktops from Windows to Linux.[1] The Israeli Department of Commerce has approved plans to switch most desktops to open source software, with similar interests being expressed by associated agencies.[2] Occasionally, these policies go beyond recommendations for procurement procedure, to encompass initiatives that subsidize the development of open source software. In 2002, as part of the plan to encourage the growth of small and medium-size software companies, a working group analyzed the France software industry and recommended that the government promote the development of open source and open standards. China is motivated to employ open source to jump-start a domestic software industry. Their Ministry of Information Industry directly funding the development of Red Flag Linux as well as ongoing collaboration with Japan and Korea to promote open source in favor of proprietary operating systems. Several other countries including Brazil, Denmark and Netherlands have also announced research and development policies with regards to open source.[3]

Various positive traits of open source such as inter-operability, customizability, avoidance of vendor lock-in, and higher security assurance has been cited as desirable qualities by open source proponents in their lobbying for explicit government promotion of open source. Whether these traits indicate that open source software is *intrinsically* more suitable than proprietary software for any government is an issue of hot debate. In an attempt to present an objective assessment of whether governmental preference should be given to actively promote open source, we explore both the arguments and counter-arguments related to the reasons often cited for this active promotion.

Note that we elect to focus on policy issues directly related to the functional roles of governments. On the one hand, governments have been rightly recognized as large consumers of software and are therefore expected by some to "lead by example" in their procurement policies, in order to have the positive effects of open source spillover into industry and society. In this section, however, we are more concerned with the *direct* promotion of open source and the issues therein.

---

[1] CNET News.com, Munich Breaks with Windows for Linux, May 2003
http://news.com.com/2100-1016-1010740.html

[2] Arutz Shevea, Israel National News, Israeli Government Moves Away from Microsoft, Dec 2003
http://www.israelnn.com/news.php3?id=54573

[3] Center for Strategic and International Issues, Government Open Source Policies, Sep 2004
http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf

Public Transparency

Proprietary software is normally distributed as object code instead of source code. This hides the internals of the software from the user. While the user usually does not require access to the internals for the ordinary operation of the software, the opaqueness of object code prevents the user from understanding the exact functionality of a piece of software without significant reverse engineering effort. Potentially, security flaws as a result of programmer error can go undetected due to this lack of visibility. More importantly, there exists the risk of software backdoors placed by malicious programmers. Growing awareness of these risks within corporations and governments alike have caused rising support for software transparency. Open source software fulfills this criterion, since, by definition, anyone is able to examine the code. As a result of the source code being easily scrutinized, bugs and security flaws in open source are routinely fixed in a collaborative manner, with numerous users contributing their time and effort to help software authors eliminate errors.

It can perhaps be argued that governments have an even greater need for transparency than corporations. Ideologically, the citizens in a democracy have a fundamental right to public information. This extends beyond the data itself, to encompass ways in which the data is actually collected, so that the public can audit and verify that the data processed, collected and distributed is not intentionally tainted or biased towards any political entity. This concept of transparency is particularly evident in grassroots movements advocating that voting machines be made open source, so that programs can be inspected to prevent the machines from performing other than their intended functions.

This suspicion of proprietary software extends into areas such as national security, which can be seen as an extension of the security requirements for software. The discovery of allegedly National Security Association (NSA) associated encryption keys in various versions of Windows have led to growing mistrust of proprietary software. Amidst the conspiracy theories, it remains unclear whether the discovered keys are to make sure that the cryptographic functions of Windows remained under US export regulations, or to enable NSA to obtain control access to arbitrary Windows computers – and without the source code, there is no way to tell if either or both are true.[4] Regardless, there has been the precedence of hardware vendors such as Netgear and Cisco hard-coding administrative users and passwords into their routers.[5] The opacity of proprietary software necessarily prevents users from easily verifying that such security flaws have been fixed by the vendors, much less detect them in the first place.

While proponents argue that open sourcing is a pre-condition to establishing the verifiability of software, critics point out that the availability of source code need not necessarily lead to better security automatically. Even in the absence of intentionally

---

[4] CNN, NSA key to Windows: an open question, Sep 1999
http://www.cnn.com/TECH/computing/9909/03/windows.nsa.02/

[5] Cisco Security Advisory: A Default Username and Password in WLSE and HSE Devices, Apr 2004
http://www.cisco.com/warp/public/707/cisco-sa-20040407-username.shtml

malicious code, security vulnerabilities inherent in software due to programmer error open wide windows of opportunities for potential misuse in open source and proprietary software alike. The availability of source code for review by its thousands of users does not necessarily imply that these audits are necessarily complete, or even competently performed.[6] Moreover, production software packages routinely come up to millions of lines in their source code, making it a daunting task to verify that software performs as intended. Regardless, proprietary software vendors have increasingly responded to calls for transparency by allowing providing licensing schemes that allow enterprises and governments access to their source code for the purposes of reference and support. Microsoft, for one, offers various Shared Source Initiative Programs,[7] which provide access to organizations, researchers, and governments that use its software. They also have a Government Security Program,[8] which does not force a country or organization to be a Microsoft customer in order to participate. These licenses allow Microsoft to retain intellectual property over its proprietary software, while allaying the consumers' concerns of transparency. Although users remain unable to modify the software with bug fixes and enhancements, it seems possible that hybrid licensing schemes such as these will start to blur the line in terms of transparency between open source and proprietary software.

Development of Local Software Industry

Due to the network effects prevalent in the software industry, some open source advocates have portrayed the commercial software model as one which favors entrenched players over newcomers. As a result, open source is seen as a vehicle that can potentially jump-start an indigenous software industry. One reason for this could be the generous licensing terms under which open source software is generally distributed. In addition to having access to working code, users are often free to make derivative works from the existing source code. This is a boon for software consulting companies, which can build on existing open source software components yet still redistribute them royalty free. Although some GPL-like licenses stipulate that the redistribution of derivative works must again be made open source, such a requirement is less of a concern when the derivative works are customized software commissioned by a customer. Studies have shown that up to 80% of the total software produced is self-developed or customized instead of off-the-shelf packaged software.[9] As such, it would seem that the extensibility of open source often lowers the barrier of entry for nascent software consulting companies.

---

[6] ONLAMP.com, Open Source Security: Still a Myth, Sep 2004
http://www.onlamp.com/pub/a/security/2004/09/16/open_source_security_myths.html
[7] Microsoft, Shared Source Licensing Programs
http://www.microsoft.com/resources/sharedsource/Licensing/default.mspx
[8] Washington Post, Microsoft to Share Source Code With Governments, Sep 2004
http://www.washingtonpost.com/wp-dyn/articles/A36880-2004Sep20.html
[9] Robert Parker & Bruce Grimm, Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts (1959-98), Aug 2002
http://www.bea.doc.gov/bea/papers/software.pdf

The effects of open source in these nascent industries have yet to be seen. It is instructive, however, to note the absence of similar policies applied to the US software industry, the largest in the world by most economic measurements. To understand this, we observe that economists routinely ask if there is a significant market failure, in order to weight the desirability of government intervention in the economy. For example, pollution occurs as a negative externality as the free market fails to capture the societal cost of emissions. Thus, the question arises: is corrective action needed to compensate for unfairness in the free market?

One point of view is that open source itself emerged to address certain needs caused by market failure. The GNU/Linux operating system is one such example, often seen as a response to the market struggles between different vendors of earlier UNIX operating systems. As a result of intense competition between vendors in the late 80s and early 90s, different UNIX offerings begin to diverge in gratuitous and incompatible ways. Technical standardization efforts took a back seat to commercial interests. Users became disheartened by this, and interest began to coalesce around GNU/Linux (as well as the BSDs and the still-unrealized GNU/Hurd), as an open and extensible alternative to UNIX. Other such examples include Apache, the dominant web server. By enforcing the HTTP standard, Apache helps to preempt divergence from that standard by providing an open reference implementation. Similarly, the Mozilla web browser (an alternative to the dominant Microsoft Internet Explorer) helps to "keep websites honest" in writing correct HTML.

For the most part, the software industry appears to be productive and efficient. According to measures, the output has risen by twenty times in twelve years, after adjusting for improvements in quality. In constant 2000 dollars, revenue increased from $35 billion in 1988 to $171 billion in 2000. The industry also appears to be highly competitive, with the prices for packaged software falling by 27 percent in the last four years and the leading firms in the industry changing places often. Of the top ten companies in 1990, five did not make the list in 2000, either due to acquisition, bankruptcy or a drop in market share. This is in contrast to the low turnover in the pharmaceuticals industry, for example, where eight of the ten leading pharmaceutical companies in the 1990 were still in the top ten in 2000.[10]

The difference between the perception of partial market failure and the healthy thriving market could be that the software industry exhibits strong network externalities. Network externalities causes a good or a service to have a value to a potential customer dependent on the number of customers already owning the good or using that service. This is found in software mainly due to the necessity for product compatibility and inter-operability. Software compatibility is important since software typically depends on other specific software which is unique and standardized. A simple example of this is that the Windows applications will not work on Mac OS X, and vice-versa. Inter-operability is significant as users often need to exchange data with each other. This can be a physical exchange of electronic documents (like on a CD-ROM or floppy disk), or virtually over communication channels such as Instant Messaging networks or email. As a result of the network effects, software companies vie for the control of the entire market since the

[10] David Evans, Politics and Programming: Government Preferences for Promoting Open Source Software, Dec 2002. http://www.aei.brookings.org/admin/authorpdfs/page.php?id=213

dominant player will attract even more customers due to its very dominance. This leads to the phenomenon of "winner-take-all" in many software areas. The control of a non-niche, large, strategically important category, such as the desktop operating systems, leads to monopolies (or *de facto* monopolies) such as Microsoft.

It is arguable whether this supply-side scale of economy can be considered as market failure. While the economics of high tech industries are less well-understood than those of traditional industries, network effects appear to be a commonplace feature of software markets. The same argument can be made for lock-in effects, where a user is essentially forced to continue buying one company's products. It appears that the government's usual degree of intervention in the industry comprises antitrust scrutiny, and it is only applied to companies suspected of utilizing their monopoly to get away with anti-competitive behavior in related markets. Open source is, on the other hand, a private party's recourse to the above-mentioned side-effects. It very much remains to be seen if any government's heavy promotion of open source would have any effects on developing the local software industry.

Publicly Funded Research

In contrast to government intervention in the software industry, government policies for open source in research are significantly more feasible since the government already actively funds public research. The connection between open source software and public research is that both can be seen as public goods in their own right.

There are two properties central to the definition of a public good. It is non-rivalrous – it does not exhibit scarcity and any one can consume the resource without reducing the amount available to others. It is also non-excludable, meaning that is infeasible to restrict its access to any one. Fire protection and national defense are classical examples of public goods. In that sense, the knowledge produced with public research has the intrinsic properties of a public good. While intellectual property necessarily imposes artificial scarcity on that knowledge in order to incentivize research, government directly funds public research due to these very intrinsic properties.

In that same way, it can be argued that open source software exhibits characteristics of a public good. The licenses for open source software typically fall within one of two categories: GPL-style licenses and BSD-style licenses. Under both types of licenses, users are free to utilize and modify the software without any restrictions or obligations imposed. While GPL-style licenses require that redistributed derivative works be must be redistributed under identical licensing terms in order to ensure openness to downstream users, BSD-style licenses have no such restriction – one can conceivably redistribute a BSD-licensed piece of open source software as closed source software. Regardless, open source licenses do away with the artificial scarcity attributed by copyright, and makes open source software behaves similarly to a public good.

This seems to suggest that open source software can be compatible with government funded research. A guiding principle in public policy is to maximize the societal benefit by minimizing the deadweight loss. In the case of public goods, this can be achieved by facilitating their distribution at as close to marginal cost as possible.

As a matter of fact, public research is already heavily involved with open source software. The TCP/IP protocol which carries all the traffic on the Internet was born in academia with an open source reference implementation, which was eventually adopted by the all commercial operating system vendors in replacement of their assorted proprietary protocols. The UNIX family of operating systems has their roots in a research prototype made from public funding. It is widely agreed that profit-maximizing firms do not have sufficient incentives to conduct long-term research, thus fundamental research in computer science has often caused spillover effects into the software industry. Open source software can be seen as a tool to produce the concrete deliverables of this innovation spillover.

By and large, there is consensus that open source software is an appropriate choice for research projects. Disagreement ensues, however, over the US government's decision to release some of its results under the GPL. One school of thought argues that since research is publicly funded, the government's responsibility is to disseminate these results as widely as possible and to help the commercialization of derivative products.[11] Licensing this software under the GPL mandates commercial products to be open-sourced too, effectively preventing commercial firms from ever claiming intellectual property over derivatives. Yet, others claim the derivatives under the seemingly "viral" GPL-style licenses can benefit the industry without being directly commercialized.[12] Existing industrial examples include embedded systems vendors, who are able to significantly reduce their acquisition costs of embedded operating systems by building on a freely available Linux kernel. In a similar vein, the Beowulf clustering software for Linux, also released under the GPL, was originally developed by NASA, and is poised to benefit hardware vendors by lowering the barriers of entry with a commoditized operating system for high availability and high performance computing.[13] Perhaps the continued openness of derivatives as guaranteed by the GPL can probably be even seen as a desirable property in itself. NSA specifically chose Linux as a platform to implement mandatory access control with their research project Security-Enhanced Linux, in part because the open development environment provided them with an opportunity to demonstrate the functionality in a mainstream operating system.[14]

Ironically, despite the wealth of research projects in the US which have been distributed under open source licenses, there is no official policy articulated by the government on the promotion of open source within public research institutions and universities. It remains to be seen whether any will be proposed and enacted, and our conjecture is that it may depend to a certain extent on the ways research software is going to commercialized in the future.

---

[11] David Evans, Politics and Programming: Government Preferences for Promoting Open Source Software, Dec 2002. http://www.aei.brookings.org/admin/authorpdfs/page.php?id=213

[12] Lawrence Lessig, Open Source Baselines: Compared to What, Dec 2002 http://www.aei.brookings.org/admin/authorpdfs/page.php?id=214

[13] Beowulf Project. http://www.beowulf.org/

[14] Security-Enhanced Linux. http://www.nsa.gov/selinux/

Banning Open Source Software

Introduction

As we have seen, open source software is a difficult issue to deal with. While it has many benefits, there are also many drawbacks. Open source software may often be less reliable than closed source software, especially when the former is produced by loose affiliations of individuals while the latter is made by corporations who have much more to lose. Additionally, much open source software is explicitly licensed without guarantees as to reliability, performance, etc. A closely related issue to reliability is accountability: when something goes wrong, will someone take charge and fix things? If things go horribly wrong, will the money you spent on the software be refunded and/or damages awarded? Traditional closed source corporate software is quite concerned with both reliability and accountability, if only because the two things affect their bottom line. Many open source developers are less concerned with reliability, and are not easily held accountable for their software.

To help ensure such reliability, most government products must be certified, to prove that they will function correctly. The certification process is a long and expensive one. While it is well within the reach of most corporations, many open source software products are unable to afford it. If the government wanted to use them, it would have to pay the certification cost itself – which would inflate the price of the software. While some private companies have recently begun the certification process for open source software on its owners' behalf, it remains to be seen whether or not this trend will continue.

There are some situations where the government may not wish for its software's inner workings to be revealed to the public at large (for whatever reason). At the same time, it is often useful to allow sharing *within* the government. For this reason, the Government Open Code Collaborative (GOCC) was created. The GOCC is an agreement between eight states to share their code with each other. It is an interesting step towards a quasi-open-source system.

Finally, the Department of Defense recently conducted a study on what software it uses. One of the issues they addressed is how much open source software was in use, and what the effects would be if it were not allowed. In the four broad categories of uses, it was determined that all of them used open source software, with the researchers relying on it the most. For many of these uses, there was no closed source equivalent. Furthermore, the actual users were categorized by level of sophistication in using the software (end-user, program writer, library writer, etc.). While the less sophisticated users could make a change to closed source software with relative ease (simply replacing Linux with Windows and Apache with IIS), more sophisticated developers were using tools like GCC and Perl which were essentially irreplaceable.

Reliability and Accountability

Two compelling reasons to avoid open source software, or ban it altogether, are those of reliability and accountability. When something goes wrong – and with software, there's always one more bug to be found – will the user be able to get support and help? This is an important thing to consider, especially for so-called "mission-critical" applications, where even an hour of downtime may cost thousands of dollars.

First, there is the issue of support. Traditional closed-source software, sold by companies, has a well-established support mechanism. Typically, one receives as much support as one pays for. The government would, presumably, be paying quite well for their software; thus, they would likely receive a high level of support. Additionally, the vendors would be sure to give the government their highest priority of service, to ensure their continual use of the software.

This area of support is one of the traditional commercial closed-source market's strengths, and one of their key selling points when competing with open source software. A Microsoft white paper (unsurprisingly) found that the cost for support and licensing for a Windows server was comparable that of a RedHat server. However, RedHat only provides support over the telephone during weekdays, while Microsoft provides it all the time.[15] RedHat, on the other hand, claims that their support is on par with and more flexible than Microsoft's, as they offer a wide array of support options for various costs.

Secondly, responsibility is an issue. Simply put, when something goes wrong, is there someone to blame? Will someone be held accountable for critical mistakes? While it should be the case that developers make the effort to make their software bug-free, the truth is that they will make much more effort if they know that they will be punished if there are devastating bugs. Much of the GNU software that Linux relies upon holds itself expressly non-accountable. The GNU Public License (GPL) states: "the entire risk as to the quality and the performance of the program is with you".[16] This would be unacceptable to many organizations. While RedHat does offer support, the fact is that many of the programs that they distribute fall under the GPL and thus the burden of the "quality and performance of the program" falls on the end-user.

Part of the reason for the limitations listed in the GPL is that many of the programs were informally developed, simply because the developers thought they would be useful. Once the program became acceptable to them, they had no particular need to continue to make improvements. In contrast, almost closed-source companies have significant incentives to make their programs function as well as possible. The cost of a critical bug is quite severe in terms of future lost sales, stock price, and reputation. Especially with government clients, whose purchases will likely be orders of magnitude larger than all but the largest of corporations, software companies will do everything they can to ensure that there are no critical bugs.

This view of open source software can be summed up by saying that many people consider it "too risky". Even some of its one-time proponents have ameliorated their pro-open-source stances somewhat. Australia, for example, had been strongly supportive of open source software. In 2002, they declared that the use of open source software is "critical for the efficient application of technology" and mandated its use in the federal government. However, this stand was recently reversed last September. In its reversal,

---

[15] http://www.microsoft.com/windowsserversystem/facts/analyses/comparable.mspx

[16] GPL version 2. Warning was converted to lowercase for easier reading in this paper.

the government stated that the decision had "stretch[ed] the industry's resources to the point that the risk of a high-profile project failure would be 'unacceptably high'."[17] [18]

The Cost of Certification

For many software products, and tools some sort of external certification on the security and performance of this software is desired to ensure the claims of the software creator. In many governments this comes in the form of certification (i.e., security tests) that a product must pass for a government to use that product. The typical case is that a government does not pay for these certifications instead the cost is pushed to the proprietary company. This is an expensive process; it is generally not within the reach of non-corporations – this is a problem for most open source software, which is generally independent. The certification is another way to build trust between a user, and a product, just as transparency is. This trust is created by using/testing the system, and trying to expose its weaknesses.

In these situations software entities which do not have the means to fund certification a couple of things can occur. The government could fund the certification and/or study of the security and performance of an open source product. This adds to the total cost of ownership for that product, and a bias to decide which of these products should be funded for certification by the government.

Another option is to have a corporation fund the certification process. Apple, whose OS X operating system uses many open source technologies, began the certification process for OS X in April 2004.[19] This should hopefully make it easier for other open source tools, and products to get certification. A distributor can fund the certification process for open source products, but this would in effect tie the government to this distributor. This limits the open source nature of the product.

Instead of having corporations directly fund the certification process it is possible for a non-profit agency to be created to help the funding of validation. This was the case of OpenSSL cryptographic libraries. A non-profit company was used the Open Source Software Institute to be the sole-source vender, and entity to validate the OpenSSL libraries.[20] This entity is funded in part by HP. While the libraries themselves have not received validation, some of the algorithms used in the libraries have been validated. When these libraries are validated it is the goal of the project to distribute these libraries across the DOD community. Motivations for this may be providing for the greater good, or by spreading the cost of certifying tools that are used by many software development companies who provide goods for the government.

Another interesting aspect of validation is typically in the US the binaries are the entities which are validated, not the source. When the OpenSSL libraries were validated a

---

[17] Dubash, Manek. Open source 'too risky' for Aussie government mandate. September 2004. http://www.computerworld.co.nz/news.nsf/0/1F6F31D95EECD55CCC256F1C0017EA92?OpenDocument&More=Platforms&pub=Computerworld

[18] Ferguson, Ian. Mandating open source "Too Risky": Government. September 2004. http://www.zdnet.com.au/news/software/0,200061733,39158367,00.htm

[19] Jackson, William. "Apple pushes feds toward broader open-source use". Government Computer News April 5, 2004 Vol. 23, No. 7

[20] http://www.linuxjournal.com/article/7644

new process had to be employed to build the source which was distributed.[21] This creates a question on whether the source should be certified, or the binary which was compiled with a certain compiler. It could be conceived that one compiler could cause a binary to be validated, while another would create a security loophole. This could be another reason why governments who require certification should not use open source solutions.

In situations where certification is required open source does not seem to be a good solution due to nature of certification. The certification would only certify a version of the product and so the highly agile and dynamic nature of open source products would not be needed. Also when an open source project is being used if a flaw is found during certification there would need to be some sort of interest to support the flaw which is exposed. This motivation may not be there without the economic rewards that come from the certification itself; mainly selling the product to the government.

Collaboration Software Groups

There may be a solution that bans open source software, but still allows a government to share information in a collaborative environment. This would allow for sharing of information within the government, but restrict information to potentially harmful third parties. The Government Open Code Collaborative is the result of such thinking.

The Department of Defense is a huge customer for the software industry, often contracting software companies to create custom software. The actual code created by these contractors becomes the property of the Department of Defense. Currently, this code is not widely shared or distributed amongst other contractors. The large size of the Department of Defense means that it can have a substantial influence on developers. If they opened up their custom code to just its current contractors, much reuse and collaboration could occur. In that reuse and collaboration, better code would be the result – hopefully creating better products in less time. This could then be further propagated to other areas of the government, allowing more reuse of code and technology. This could create an atmosphere similar to the Apollo Project, allowing technologies created for one government application (e.g., the moon landing) to be used to better the country as a whole.

In June 2004, an organization was created with that goal in mind. The Government Open Code Collaborative[22] was created by government agencies in eight states (New York, Massachusetts, Rhode Island, Pennsylvania, Utah, Kansas, Missouri, and Virginia). It is an effort to allow government agencies to share code within their community. (The definition of "community" in this context is United States governmental agencies, not the people within the states themselves.) The basic idea is to allow anyone who is in the community to use any code published in that community, but not allow outsiders use of the code unless specifically condoned by the author. This allows sharing of information and technology on a more restrictive community level instead of an expansive global level, advancing technologies in these sectors without the fear of exposing information to inappropriate third parties.

---

[21] http://www.linuxjournal.com/article/7644
[22] http://www.gocc.gov

Case Study: Department of Defense Investigates Banning Open Source

In 2002 a study was conducted by the MITRE Corporation as to the impact of how much open source software is being used in the Department of Defense and why it was in use. This study also proposed a hypothetical question: "what would happen if open source software were banned?" This gives an insight on the possible outcomes of a government banning, or disapproving the use of open source software. According to the research this situation "has a real-world analogy in the form of proprietary licenses that if widely used would effectively ban most forms of FOSS."[23]

The study discovered that open source software is generally used in four categories: Infrastructure Support, Software Development, Security, and Research. There were 115 applications used total that were considered open source; of these, 65 were used in Infrastructure Support, 53 in Software Development, 44 in Security, and 21 were being used for Research (different departments using the same application only counted once for the total application count).[24] For many of these systems, it was determined that there was no equivalent closed source alternative where an open source product was being used.

This lack of a closed source equivalent was especially true in three of the areas: Infrastructure Support, Software Development, and Research. With Infrastructure Support, it was stated that "the strong historical link between FOSS and the advent of the Internet means removing FOSS applications would result in a strongly negative impact on the ability of the DoD to support web and Internet-based applications" – not surprising, since the Internet was almost exclusively founded on, and continues to rely on, open source software.[25] For the Software Development community, the study stated that most of the tools used to create software – compilers, interpreters, debuggers, etc. – were open source software; without those tools, the development community would be highly restrict in their ability to create software. Lastly, Research was an area where many of the tools used are so cutting edge there is no closed source project that is available (and, often, their open source project is unique). This is because either there is no market for this product, or it is so cutting edge that the market has not yet been realized.

It is interesting to note that open source software was being used in research not just to help in the research effort, but also to jumpstart research by publishing code that could easily be expanded upon by other researchers. These researchers would be able to quickly and cheaply work on ideas relating to the open sourced code, without having to reinvent the research solution already proposed or to wait until it became a commercial solution.

In addition to being divided into groups by area of work, software users were also divided up into groups based on how they used the software. There were four categories:

---

[23] **Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense** by The MITRE Corporation http://learn.arc.nasa.gov/worldwind/dodfoss.pdf

[24] Ibid.

[25] Ibid.

operational users, basic code development users, advanced code development users, and sponsors.

The operational users just used the software as-is; they would almost never look at the code, let alone modify it. Examples of products that were typically used in this manner are Linux and Apache. The study determined that these users would not be greatly affected if open source was banned as long as a proprietary system was in place to replace the open source system. Since none of the users was using open source software for anything particularly esoteric, there exist commercial equivalents for almost all of their needs.

The second type of users primarily did basic code development. These developers created code which was executed or compiled with open source tools. These tools included Perl, JBoss, and GCC. The study observed that these tools must ensure that the code created would not fall under the license of the producing tool, which would (in these cases) force the code itself to be open source. This would, as may be imagined, be a problem for some defense applications. However, little of the software being used had such a "viral" licensing scheme. On the whole, the developers rely heavily on open source software; if these tools were banned, viable alternatives would need to be found to replace them.

The advanced code developers were one of the smallest categories of developers. This group developed libraries, tools, and other large creations which were added to open source software. The benefits of using an open source model were not described in this paper; however, empirical evidence suggests that they include helping to propagate research and development, and gaining wider use and acceptance of a product, among other things. The public at large also benefited, of course, but this was not of particular interest to the Defense Department. If open source was banned, this would create a need to create avenues to distribute code within the scope of the Department of Defense.

The final category is that of open source sponsors. Prominent members include the NSA (sponsor of SELinux) and MITRE (sponsor of Collaborative Virtual Workspace). These sponsors are trying to advance software in a particular field or community that they're interested in, and feel that open source is a good fit for them (for a variety of reasons). By contributing money to a project they're interested in, these sponsors can help advance a field of development that is stagnant or in its infancy. If open source was banned another system to develop and foster research would need to be created; either allocating more funding to research, or giving companies incentives to research certain areas which the DOD are interested in.

The study found that open source projects were vital to many important aspects of the Department of Defense, a conclusion which surprised the paper's authors. For security purposes, open source products were often used because of their quick bug fixes and frequent updates. It was also found that many important security-related tools were developed by open source communities, to both find and analyze security holes. These include products such as SNORT, and SARA.

The study concluded that completely banning open source projects will ban many of the technologies that the government is now using. This will cause considerable migration cost, as well as security issues as an old, well maintained system is traded in for a new and unfamiliar one.

Deciding between Open and Closed Source

Introduction

There are multiple reasons for governments to promote open source software. Yet this promotion does not go without risk. Many countries recognize that there may be benefits from using open source software, so the government's policy is that public authorities should consider open source software *alongside* proprietary software. This means that each government IT initiative should make assessments and do cost-benefit analyses to determine which whether an open source model, a closed source model, or a hybrid of the two is correct for the project.

There are many factors that should be considered in making such a decision. We present here some of the major ones that governments should consider. While most open source software may have no licensing costs (unlike almost all closed source software), they may not necessarily be cheaper. The Total Cost of Ownership (TCO) must be considered, as both open and closed source software may have hidden costs above and beyond the initial license. The choice between open and closed source needs to be considered above ideological considerations – which one allows for more user productivity? In many third-world countries, software piracy is a serious issue. Could open source help reduce this? Both open and closed source software are helping to make a public good and to bridge the digital divide, but each are doing so in radically different ways. Some countries may consider vendor lock-in and a lack of market competition as a bad thing; others may see these factors as good because they benefit their domestic software industry. Finally, a government needs to consider how important open standards are, and whether the software they're considering – open or closed – supports them.

Does Open Source Software Reduce the TCO?

The Total Cost of Ownership (TCO) is a hotly debated issue in the closed-source versus open-source debate. Much open source software is given away at no cost. One could thus naively conclude that this software is completely free. However, this would be overlooking the cost required to train users for the software, the costs of maintenance, the cost of bugs, etc. Such factors apply in the closed source software world as well. As such, comparing the cost of a hundred dollar Windows license against a free Debian license should not simply end there; not just the initial licensing fee but the Total Cost of Ownership must be considered.

The TCO is an important factor for every project that is initiated and implemented. Some projects, such as the California Department of Transportation (Caltrans) have found success with the open source software model as a less expensive alterative to proprietary software. Caltrans initiated a now successful project for identity and password management. The non-open source solution was quoted at $500,000;

Caltrans chose to implement a Linux based open source solution for $220,000 – less than half of the projected closed-source cost.[26]

Other projects have not found the same successes in reducing the TCO by using the open source software model. The Red Escolar Libre (Free School Network) Project in Mexico is an example of an OSS project that increased the TCO. The goal of the project was for 120,000 schools in Mexico to each have one server networked with six desktop computers. The initial software estimate using Microsoft software was $830 per school. The project leaders decided not to pay licensing fees when other software could be obtained for free. But the support for the implementation and use of the OSS was much higher than anticipated and Mexico found that the TCO would have been less if they had gone with Microsoft. Mexico is now considering entering into an agreement with Microsoft to finish the project.[27]

In a letter to the Peruvian government, Microsoft observed that open source software is not "free of charge". They say that "research by the Gartner Group (an important investigator of the technological market recognized at world level) has shown that the cost of purchase of software (operating system and applications) is only 8% of the total cost which firms and institutions take on for a rational and truly beneficial use of the technology. The other 92% consists of: installation costs, enabling, support, maintenance, administration, and down-time."[28] Thus, governments need to conduct studies and determine the actual cost of the on-line system, making an educated decision not based solely on the licensing fees; while the initial cost for open source software may be less, the overall cost to support it may be more.

Which Leads to Better End-User Productivity?

In choosing software, it is important to evaluate whether or not the chosen system is actually meeting the requirements of the project. Most software used by governments is simple word-processing or budget-balancing applications that require only basic knowledge of the operating system. The goal of these systems is to increase end-user productivity. The governments must evaluate if choosing an open source alternative such as Linux enables that end-user to be as productive as a proprietary solution could be, or whether the focus-group-tested user interfaces of Windows or Mac OS X are superior.

In 2003, a Berlin based firm conducted a feasibility study to determine the risks of using Linux in public agencies. The study compared Windows XP and SuSE8.2 with KDE 2.1.2 in performing office tasks. The participants chosen had no prior knowledge of the systems and were separated into two groups; both groups performing the same task on the assigned operating system. The results showed that using Linux for desktop tasks is certainly feasible. But on average, it took the Linux group longer to perform the tasks than on a Microsoft system. The study also determined that Linux has considerable problems with its user interface; for example, the wording of applications was poor and

---

[26] California Performance Review. SO10 Explore Open Source Alternatives. http://www.report.cpr.ca.gov/cprrpt/issrec/stops/it/so10.htm#2b

[27] Brod, Cesar. Free Software in Latin America. Jan. 2003 http://www.brod.com.br/files/helsinki.pdf

[28] González ,Juan Alberto. Microsoft's "Fear, Uncertainty and Doubt" (F.U.D.) letter to Peru concerning free and open source software. March 2002. http://mirror.opensource.dk/docs/msFUD_to_peru.php

the desktop interface was not intuitive. The study should be considered by any group considering the use of open source software.[29]

How to Decrease Software Piracy?

Pirated software has become a large issue for developing nations. Figures from the Business Software Alliance (BSA) and the World Bank Development Indicators database 2001 show that software piracy is prevalent among developing nations: 97% of software is pirated in Vietnam, 92% in China, 83% in Pakistan, and 70% in India.[30] [31] [32] Countries are attempting to combat the high levels of piracy primarily to avoid US trade sanctions and the potential loss of international business. Additionally, many of the developing nations want to be recognized as a respectable country among the first world.

Open source software gives 2nd and 3rd world nations a chance to be labeled as something other than "pirates". Open source gives these countries a way to legally enter into the IT industry. Pakistan has created a large e-governance initiative to become legal; the software used by the government must be legal. Pakistan also has an effort underway to equip rural schools with computers. This effort is to be completely legal and high profile, so that 1st world countries can see their progress.[33] There are many other nations, such as Peru and Vietnam, creating similar projects for the same reasons.[34]

For its part, Microsoft is also trying to reduce the amount of piracy among developing nations. Their model is to work with each government on an individual basis and create pricing models that suit the economic needs of the country. For example, they offered to sell their Office suite for $36 rather than the standard $300 in the US.[35]

Which Helps Make Software a Public Good and Bridge the Digital Divide?

Providing technology and computer literacy to the world is an important issue. Developing nations need software to be a public good in order for this to happen. Open source software inherently benefits the public because it is publicly distributed. It is also beneficial because it is nearly always free (as in beer). It is thus more readily converted to a public facility because of the lack of licensing fees, and because it may be more easily modified. For example, the Simputer (Simple, Inexpensive, Multilingual People's

---

[29] Relative User Experience Architecture. Linux Usability Study Report. Aug. 2003 http://www.linux-usability.de/download/linux_usability_report_en.pdf

[30] The Indian Express. India sits on the fence in open source debate. April 2004 http://www.indianexpress.com/full_story.php?content_id=45825

[31] News Forge. Talking with Open Source advocates from Peru and Vietnam. Oct. 2002 http://www.newsforge.com/business/02/10/23/0049208.shtml?tid=19

[32] Noronha, Frederick. Open Source, Free Software Opens New Windows to Third World Computing. April 2002 http://www.school.net.th/linux/news/linuxpakistan/

[33] Noronha, Frederick. Open Source, Free Software Opens New Windows to Third World Computing. April 2002 http://www.school.net.th/linux/news/linuxpakistan/

[34] News Forge. Talking with Open Source advocates from Peru and Vietnam. Oct. 2002 http://www.newsforge.com/business/02/10/23/0049208.shtml?tid=19

[35] Dravis, Paul. Open Source Software Perspectives for Development. Nov. 2003 http://www.infodev.org/symp2003/publications/OpenSourceSoftware.pdf

Computer) was developed by the Indian Institute of Science to bridge the "digital divide" between the rich and poor. Southern Indian state of Karnataka is using the handheld Simputer to automate the collecting of information about the crops cultivated in their jurisdiction, hoping to reduce the information gathering time from one year to one month. In this case, the village accountants possess the Simputer and collect the data from the farmers. The Simputer is a handheld computer that stores data via smart cards, is networked via wireless modems, and enables voice to data conversion for the illiterate. The Simputer uses OSS to support the above functionality because the government could not afford to build an operating system from scratch and it needed to modify the operating system in order to support the device. Given this, open source software would be the only viable option.[36]

Microsoft is also committed to bridging the digital divide. Microsoft has recently entered into an agreement with the United Nations Development Programme (UNDP) to work to create community centers and work with governments to provide basic computing services to some of the world's most underdeveloped nations. Microsoft has set aside $1 billion over the next 5 years to fund this project.[37]

Is Vendor Lock-In and Market Competition a Concern?

A government may want to increase their technological self-reliance by lessening their reliance on external providers and by increasing the local work force, thus reducing vendor lock-in and increasing market competition. This decision is often motivated by the hope to increase software production within its own nation, thus increasing its national GDP. One example is the city of Munich, Germany, which is scheduled to migrate 14,000 existing systems in the public administration to open source software solutions (using Linux as the OS). The mayor stated that the rationale for deciding to use open source software was to "set a clear signal for greater competition in the software market."[38]

On the other hand, the United States is a good example of a nation that would not necessarily be interested in promoting greater market competition. The International Institute for Educational Planning (IIEP) cited that "US based firms generated 56% of the revenues and 96% of the profits from the global IT industry."[39] Hence, the US is already technologically self-reliant. Furthermore, any move to open up the market would likely result in more international competition, which would hurt the US dominance of the software market.

Reducing vendor lock-in may also motivated by the ability to choose each part of a solution (hardware, software, support) rather than being forced into an end-to-end solution. One example of this is the Electronic Government Initiative in San Paulo,

---

[36] The Dravis Group. Open Source Software Case Studies Examining its Use. April 2003 http://www.dravis.net/images/Open%20Source%20Software%20(Dravis).pdf

[37] UNDP and Microsoft Announce Technology Partnership To Combat Poverty in Developing Nations. January 2004. http://www.microsoft.com/presspass/press/2004/Jan04/01-23WorldEconomicForumPR.asp

[38] Dravis, Paul. Open Source Software Perspectives for Development. Nov. 2003 http://www.infodev.org/symp2003/publications/OpenSourceSoftware.pdf

[39] Free Open Source Software for E-learning. June 2004. http://www.unesco.org/iiep/virtualuniversity/forums2.php?queryforums_id=5&querychapter=1

Brazil. The e-government has set up 72 functioning telecenters that provide free computer use and Internet access to the people. Each center costs $10,000 to set up; all are using open source software and diskless workstations. The decision to use open source stemmed from the fact that the project was able to "acquire computers with less hardware and power requirements than required by a Microsoft solution" – a clear example of reducing vendor lock-in.[40]

How Important are Open Standards and Interoperability?

The European Commission (EC) proclaimed at the "Open Standards and Libre Software in Government" conference (held in November, 2004) that open standards are of utmost importance for inter-agency collaboration. The EC has created the European Interoperability Framework document, which supports a strong link between open standards and open source software. It says that "OSS products are, by their nature, publicly available specifications, and the availability of their source code promotes open, democratic debate around the specifications, making them both more robust and interoperable. As such, OSS corresponds to the objectives of the Framework and should be assessed and considered favourably alongside proprietary alternatives."[41]

The open source community is not the only software developers adhering to open standards. The XML open standard is one such example. Nearly all of the major proprietary database corporations, such as Oracle and Microsoft, support the XML open standard.

It is also important to note that there can be reasons a proprietary product may not want to support open standards. If a product has found a more efficient way to accomplish the task that an open standard does, the company behind the product may choose not to support the open standard in order to try to gain a portion of the market share from the increased efficiency. Additionally, if a product has a majority market share, it will naturally wish to discourage open standards to prevent the rise of competitors. Since the competitors could not interact with the *de facto* standard, they would have diminished network effects.

Conclusion

Open source is a new and trendy way of developing software. While it has proven to be surprisingly successful and have many benefits, this does not make it a panacea. There are many factors to be considered in formulating a policy on open source; concepts ranging from philosophy to economics present viable arguments. In the end, it is up to each individual country to examine its circumstances and decide what the correct course of action is. It seems likely that an appropriate decision would be the middle ground – open source should not be banned, but neither should it be mandated.

---

[40] Dravis, Paul. Open Source Software Perspectives for Development. Nov. 2003 http://www.infodev.org/symp2003/publications/OpenSourceSoftware.pdf

[41] Ghosh, Rishab Aiyer. EC announces Open Standards Definition. Nov. 2004 http://www.newsforge.com/article.pl?sid=04/11/19/148236

The Australian government's retreat from its mandate of open source software shows that, for all of its virtues, open source software does not have all of the answers. On the other hand, the Department of Defense and the Internet itself provide compelling evidence that open source software should not be heedlessly discarded. The middle ground is not completely satisfactory, but most of the time it is the best of the possible options. This is a complicated issue, and is sure to generate much controversy in years to come.