Opening the Open Source Debate

A White Paper
June 2002

…Like a map to a buried treasure, once you have the code, you can create the software…

Kenneth Brown
President
Alexis de Tocqueville Institution

Opening the Open Source Debate

Table of Contents

Opening the Open Source Debate

By Kenneth Brown

I. In the Beginning….

What we know as software is the end result of painstakingly typing thousands, sometimes millions of lines of text in programming language (see Appendix 1). This text is commonly known as "source code." The source code is unrecognizable to most unless you know something about programming, but this text is the "secret formula", the underlying schematic that enables each process and function of a software product to work.

The open source debate is about keeping secrets. Completed (written) software is often locked by its programmer, hiding the underlying code from its user. Software can only be modified in its "unlocked" state when the source code is viewable. Software's locked state is also described as its "executable" format. Executable software is commonly sold in stores and available commercially. Executable software accompanies binary code also known as machine code. This binary code is readable by the host PC and used to mechanically operate the software.

Developing software inherently creates two versions – one that is sealed or executable and one that is open source, disclosing the underlying code. The decision to keep code secret is the prerogative of the programmer. And it is this prerogative that propels us into the open source debate.

Open source software is not necessarily free software. Programmers sell their software as completed products, with or without the code. Likewise, open source and non-open source products are both sold and available free to users. For example, Apache is an open source web server. Apache accompanies its source code, but it is also used to create commercial applications. Conversely, Adobe Acrobat Reader, a widely used proprietary software for viewing documents is free, however, its owner Adobe has no intention of releasing its source code.

Like a map to a buried treasure, once you have the code, you can create the software. Programmers that make a living leveraging the unique value of their software, do whatever it takes to keep their code secret. As expected, most successful programmers and companies do not disclose their code and sell their software without the source code. This is commonly referred to as selling proprietary software. Today, there are several variations of proprietary software produced and sold around the world (See Appendix 2).

Programmers have freely exchanged source code for many years. Researchers and scientists promoting their findings would often publish their code. Today, programming enthusiasts have hundreds of clubs, associations and e-mail discussion groups to discuss developments in programming language and new applications. As expected, many of these programmers are able to troubleshoot problems by exchanging their code with one another in these groups.

While individuals within each of these groups exchange code for various reasons, completed software often comes with rules for using, copying, selling and distributing the respective code. Depending on their interests, programmers choose agreements for their products they feel most comfortable with. Today, there are least 30 different licenses (See Appendix 3) with a number of variations (see Appendix 4) that programmers can choose from to distribute their open source products.

II. GPL Open Source – The Gift That Keeps Taking

Most open source licenses allow users to freely integrate code into proprietary software without too much difficulty. For example, the MIT license (see Appendix 5) is one of the more lenient licenses and has very few requirements. However, the license called "general public license" (GPL) is the opposite (see Appendix 6). The GPL is one of the most uniquely restrictive product agreements in the technology industry. As expected, the controversy of the GPL is rooted in the language of its license.

Its preamble outlining the most major difference from all other licenses begins with the rule that, "… if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights…". The license includes these details and others explaining that if you include GPL source code, the new product also is GPL. Section 6 makes this point very clear stating, "… Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein….".

The GPL requires that if its source code is used in any type of software product (commercial or non-commercial) for any reason, then the entire new product (also known as the derivative) becomes subject to terms of the GPL open source agreement. For example, if the code for a software application was originally 10 lines, and 5 lines of GPL open source is added to it, then the entire 15 lines becomes GPL open source. In effect, if a programmer uses GPL open source in a proprietary product, he agrees that the new product can be changed, modified and distributed as freely as if it were purely open source. Thus, the question becomes, "why would a programmer that sells his product as proprietary software agree to incorporate GPL open source, in effect allowing its source code to be indiscriminately distributed?" The answer is -- most don't! This is why there is such heated debate over the GPL. David Wheeler, technology expert in Virginia on open source and proprietary source comments, "without licensing the source code in a multi-license format, (referring to other more permissive licenses), it is very difficult for GPL to work for a proprietary business model…"

The origin of the GPL license provides a helpful perspective to understand its approach. In 1971, Richard Stallman joined the Artificial Intelligence Lab at the Massachusetts Institute of Technology in Cambridge, Massachusetts. His job was to improve a computer operating system known as the Incompatible Time-Sharing System (ITS). Stallman was well known for his negative opinions about proprietary software. In a conflict over building enhancements into an operating system called LMI, Stallman reverse engineered a rival operating system marketed by Symbolics to help LMI build an application to work with the system. Shortly after the LMI incident, Stallman resigned from the MIT faculty to build a free open source Unix-like operating system. He named the project GNU, which stood for "GNU is Not Unix."

In 1989, Stallman decided that the open source community should be more organized and founded the Free Software Foundation (FSF). FSF and Stallman evolved the open source discussion into an advocacy group, promoting the idea that all software should be free. FSF became well known for its position of free software as well as its radical ideas to end patents on inventions. Stallman's group was determined to convince (whoever would listen), that individual ownership of ideas adversely affected technology.

Stallman's Free Software Foundation introduced the General Public License (GPL). The GPL became referred to as "copyleft", aptly describing its license as "all rights reversed". The viral property of the license enabled developers to copy, examine, modify, and redistribute source code freely. Users are free to charge for the cost of distributing GPL products. All derivations of GPL code became freely available to anyone that wanted to use it, regardless of the purpose.

The copyleft concept moved from FSF to other organizations. However, as other organizations began to form, support for FSF began to erode. The controversial nature of Stallman's position began to turn away his supporters. By the early 90's, open source enthusiasts began to view Stallman as an extremist and fanatic. The rise in the popularity of Linus Torvalds and the Linux open source operating system began to create new supporters. Ironically, Linux supporters became the biggest proponents of the GPL. Although Stallman is a fallen hero in the open source world, most open source products today are distributed under the GPL license.

In Wheeler's paper, More Than a Gigabuck: Estimating GNU/Linux's Size, Wheeler documented that in 2000 almost 60% of all open source is licensed under the GPL (see Appendix 7). Wheeler comments, today I would be confident that the number has probably grown to 80%." All open source is not GPL, however, the surge in GPL'd open source makes a substantive part of the open source debate about the GPL.

III. The Myth of a "Public Software" Community

Widespread support for GPL open source lies in the IT community's frustration with competitive, closed proprietary software. But in fact, it is quite common that programmers experiment with open source until they see an opportunity to capitalize on an idea, then embrace proprietary standards. One could joke that open source has been a bridesmaid but never a bride. The story of the web browser is an example of this reality.

[1]In an essay entitled, "Storming the Gates", by Nathan Newman, the Internet's ironic history is well documented. President Eisenhower founded the Advanced Research Projects Agency (ARPA). ARPA coordinated research and development on certain military projects and funded a number of college and civilian agency projects. Project MAC at MIT was a government-funded project with $3 million per year to create "time-sharing" computing minicomputer technology (later the home of Richard Stallman). ARPA also funded the Augmentation Research Center (ARC) at the Stanford Research Institute, as well as the National Center for Super-Computing Applications (NCSA). By the early 60's, ARC was experimenting with e-mail, word processing

[1] Storming the Gates, The American Prospect Online, March 27, 2000. Material substantially rewritten for use in this paper.

software and hypertext linked documents. ARPA led to an expansion of experts that would spur growth of the Internet.

Project MAC, ARC, and other ARPA-funded institutions formed the collaborative research network that would eventually become the first Internet. In 1969, the first four "nodes" on the network were linked together. In October 1972, ARPANET was publicly demonstrated. ARPA presided over the creation of tools to use the Internet such as the TCP/IP protocol and the File Transfer Protocol (FTP). At this time, the government was encouraging a wide variety of firms to make these kinds of improvements. A number of firms that foresaw the potential of the Internet began pouring millions of dollars into research and development concepts. In the 1990's, one company in particular mirrored the nexus between open source and proprietary software development.

Jim Clark, the founder of computer maker Silicon Graphics and expert on UNIX standards, founded Netscape, and set out to compete directly with the public domain product Mosaic. Netscape became one of the first companies that created proprietary software from the original ARPA funded research. The first Web "browser," Mosaic, was part of a larger federally funded project to create graphics based collaborative tools. Mosaic was first introduced on the UNIX platform in January 1993, and debuted on Macintoshes and PCs eight months later. Copyrighted by the University of Illinois, Mosaic was an open source product and could be downloaded for free by individuals and companies wishing to use the Internet for internal communications. Through a commercial partner, Spyglass, NCSA began widely licensing Mosaic to computer companies including IBM, DEC, AT&T, and NEC.

Jim Clark hired Marc Andreesen, a member of the Mosaic team, along with a number of other NCSA programmers, to design what the company called, in-house, "Mozilla, the MosaicKiller." Within six months, Clark's team succeeded in creating a powerful browser with additional features and capabilities to surf the Internet at faster speeds than Mosaic. Netscape also went beyond the HTML standards embedded in the Mosaic browser and included the ability to display text formatting, a feature that didn't exist in the NCSA version. This made Netscape more attractive to users than Mosaic. However, it also meant that web pages designed to work with Netscape would not be readable by other browsers.

Netscape was an aggressive firm. It endeavored to make its web browser the proprietary standard for web access, hoping that it would inevitably become more important than the PC operating system. Netscape began distributing its browser free to users. This strategy all but eliminated interest in Mosaic and NCSA led standards. Because Netscape was also able to do this without paying licensing fees to NCSA, it was able to undercut other commercial browser companies that had to meet NCSA license fee requirements. Not only did Netscape crush competition with its free browser model, but it also infuriated members of the open source community by aggressively introducing proprietary standards to the public Internet, something they felt no one should own. Conveniently, Netscape turned its enemies to Microsoft and their new browser, Internet Explorer.

Newman's paper and the Netscape story however prove that public private partnerships are valuable to technology efforts. While government and research institutions are more efficient in long-term projects, the private sector is more efficient with creating solutions quickly. Conversely, it would be impossible to expect the efficiencies of the private sector to appear in the model of the public sector, just as it would be inane to expect the government to operate like a business. However, it does make sense for each to respect the interests of the other and work together.

IV. The Government and the GPL

The development of the Internet is only one example of prodigious partnerships between federal agencies, academia and private enterprise. Government sponsored research works well with open source, particularly because open source is intended to be perpetually improved, creating an ideal source for innovative ideas. However, the government's use of GPL'd software is a very different scenario.

A government agency that chooses to use GPL'd software is subject to the same restrictions that a private sector firm is, which creates a number of new considerations. David F. Skoll, President of Roaring Penguin Software commented, "Yes, the government should use and distribute GPL'd software. If the government uses GPL'd software and distributes binaries, it's obliged to distribute the source code." While there are tradeoffs for use of the GPL in the private sector, there are a number of unique considerations for the use of GPL'd software by federal agencies.

The most immediate consideration is security, particularly because the GPL has strict rules for any software that has been or will be distributed. To this extent it is important to closely scrutinize the impact of the government's decision to use software that is widely available, in detail, to an undisclosed community. Inherently, the closed nature of proprietary software is its first line of defense regarding security issues. Conversely, a federal agency's decision to use GPL open source from the public domain must accompany the assumption that potential attackers either already have or could easily acquire a duplicate copy of the same exact source code used by a federal agency. This is relevant because if the software is in the public domain, then potential hackers also have had the opportunity to study the software closely to determine its vulnerabilities. A second factor which makes the government's use of GPL open source questionable is the model which encourages Internet dialogue to acquire improvements, patches and fixes to the software. This process should be closely analyzed as well to determine if these encounters pose any type of security issues.

At the center of the security question is the debate over whether the availability and exchange of code is even a security issue. Wheeler comments, "There are many programs developed by the government which are THEMSELVES classified, and many – and probably most – of the various programs most important to national security are in this category e.g.,, weapons systems). In that case, neither binaries nor source code of those particular applications are released to anyone else; besides being illegal, releasing the binary executables would give away far too much information."

Since many federal agencies do classify their source code, the remaining question is whether there is a reason for agencies without top secret activities to keep their source code secure. Jason Rexilius, an ex-USAF intelligence analyst currently an applications architect comments, "There could be many reasons for classifying or keeping secret code, some valid some not.…One might be that the algorithms used in the code would be classified. This is common in the private sector and is a valid reason for keeping certain source closed. As an example, an algorithm developed by an auto manufacturer used for testing shocks may be kept as a corporate secret rather than patented because patents are in public domain and it would be difficult to prove that a competitor was using it."

Many experts contend that keeping code secret (security by obscurity) is not a fool-proof approach. Proponents for opening the code argue principally that instead of depending on the proprietary closed code of one vendor, the government would benefit from a) substantially increase its pool of talent to fix bugs, etc. and b) get fixes to security problems faster. However, it is important to note that this decision assumes that the tradeoff will always be cost-benefit. Specifically, it assumes that opening the code to potential attackers is an acceptable cost to receive the benefits of more help and faster fixes, because it is a tradeoff, the costs must be thoroughly analyzed.

The first consideration is the determination of the value or utility of of the source code to potential attackers. Rossz Vamos-Wentworth a programming expert, comments, "If the government uses GPL software, the government is to release their version(s). If the software is related to security, it really doesn't matter if the code is available or not. Security holes are eventually found, with or without open source code. If the security software is well done, having the source code will not make it easier to "crack"."

However it is important to note that opening the code to potential attackers provides a free education, inadvertently teaching attackers how to create and mimic the same security techniques. Andrew Sibre, a programmer with over twenty years of experience insists, "Having a license for binaries only gives you a "black box": you don't know what it's doing, or how, unless you want to go insane trying to reverse-engineer it with a debugger (illegal under the term of most licenses…) Having the source lets you see what it's doing, how it does it, and permits you to modify it to meet your particular requirements (including security related ones)."

To this extent, government officials should be concerned that providing too good of an education could lead to an adversary cracking their system. Outlining a noteworthy scenario, Rexilius comments on how this information is later exploited in a closed system, "…You have a closed network in your building with guards, cameras, and ID badges to prevent physical access. You also have proper security INSIDE your network utilizing strong passwords, encryption, and sound well-built applications. Now if you open the blueprints for every aspect of it to the world, your adversary can reconstruct a test lab in which he can create tools he may need. He then manages to gain physical access to your network by posing as a janitor and has a 5 minute window in which to try and crack your system. If you have a well-built system this will not matter, however if you have a security problem and the attacker had pre-built his tools, this would be the window he needed…"

While executable software without source provides the binaries, de-compiling techniques can provide much of the source code. With scenarios that point to the value of code, the question for government officials is not just a technology question but a policy one as well. What should be the policy regarding the release and availability of code? Is one set of code more of a security risk than the other? Wheeler comments, "If source code is the "blueprint", then the executable (binary) is the "building". Why would an attacker need a blueprint, when they can get an exact copy of the building to explore and search for vulnerabilities? And in the world of software, an attacker can automatically generate the blueprint from the building. Protecting the blueprint, but not the building makes little sense."

Another consideration that deserves further study is whether government officials should openly exchange source code in discussion groups and community bulletin boards on the Internet. The bulletin boards are fluidly used to alert other programmers about the need for a particular patch to their software. Again, the issue would demand further study, but it is unquestionable that an attacker looking for an opportunity to exploit weaknesses in government systems could track messages on the Internet. It would be efficient because they would have a party to build trust with, would be alerted immediately about the status of the problem, and would be able to receive source code and intimate information about the system without ever providing a fix. The discussion boards on the Internet are an excellent tool for seeking and obtaining patches however, this process could be used to inform someone with bad intentions about new opportunities to exploit a government's IT system.

Carely Lening, student at Pierce Law Center in Concord, New Hampshire offers a simple analysis of the dilemma: "This all boils down to a matter of trust. If you trust the source boards and volunteers, the patches can be a great asset. Provided people don't have bad ends, the immediate, large aspect of the "downsie" (drawback) only comes when the community either a) doesn't check their work or b) doesn't have very high morals…"

Moving to another point, if someone decided to create a backdoor that could later be exploited, the Internet discussion boards would be an ideal way to fool someone into downloading a Trojan horse. An unwary user could download the fix that could later be used by the attackers to exploit the system. The potential for holes and back doors in open source exchange on the web would require that federal officials have the resources to thoroughly scrub any code exchanged on the Internet.

These concerns do NOT dismiss the effectiveness of the model. A number of programmers have achieved well-deserved notoriety by combing the discussion boards to take on the challenge of fixing software glitches. However, due to heightened attention to security, any and every scenario that would foster a breach of the government's IT infrastructure becomes a valid concern.

For example, if the Federal Aviation Agency were to develop an application to control 747 flight patterns from a widely distributed GPL open source code, security questions would include: Would it be prudent for the FAA to use software that thousands of unknown programmers have intimate knowledge for something that has been actively targeted? Could the FAA be sure that intimate knowledge of this software is not being shared with the wrong parties? Would widespread awareness that software the FAA chose to use originated in the public domain hasten an invitation to hackers?

Needless to say, the aviation agency is not the only department that would have these types of security concerns. The Securities and Exchange Commission, the Internal Revenue Service and the Federal Reserve all monitor, collect and store sensitive information. In each case, the potential for disrupting the IT infrastructure at any level in the government should be rigorously studied.

A final consideration for the government is its productive alliance with private enterprise. Its decision to use GPL source code will inherently turn away many of its partners in government sponsored research projects. Security, as well as other impracticalities make GPL open source very unattractive to companies with concerns about using the research for profitable enterprises. Many GPL supporters are vehemently against intellectual property rights such as copyright and patent protection of software. Thus, the government decision to embrace GPL may be at the cost of ending its relationship with the intellectual property based sector.

Immediately, this would limit the number of qualified vendors the government could choose from to deliver products. A worse consideration is that the use of the GPL could inadvertently create legal problems. IP community members could argue that the government's choice for GPL open source is restrictive and excludes taxpaying firms from taxpayer funded projects. This creates another dilemma for the government because its use of GPL could inherently interrupt the flow of technology from research projects to the private sector. Without value, the government's funding for research would inevitably be adversely impacted.

<u>V. "Intellectual Property Left"</u>

U.S. intellectual property (IP) statutes have been a beacon for inventors around the world. The U.S. model for motivating, compensating and protecting innovators has been successful for almost 200 years. Proponents of the GPL seek to directly compete with the paradigm of intellectual property, thus it is vital to analyze this phenomena as well. Rossz Vamos-Wentworth's ideas reflect the sentiment of those looking for radical changes to the status quo commenting, "Intellectual property is a relatively new concept. I find it difficult to support the idea of someone owning an idea…In my opinion, intellectual property rights are out of control and need to be reigned in. Currently, IP rights for the most mundane concepts are used to stifle competition and growth…"

The question of reverse engineering is probably one of the biggest conflicts between the IP and the GPL open source community. The largest pool of software is still proprietary software. To keep GPL products relevant and up to date, GPL enthusiasts perpetually reverse engineer closed source and proprietary software to achieve interoperability of open source products with proprietary products as well to encourage the continued use of open source products.

For example, many hardware manufacturers (also known as original equipment manufacturers or OEMs) keep hardware specifications secret, which includes the accompanying software that enables the product to be interoperable and work with other products. An OEM's new printer accompanies software known as a "driver". Once loaded, the driver enables the PC to communicate with the new printer and print documents. Open source enthusiasts reverse engineer the driver and make the code available in the public domain. Once a driver is reverse engineered, programmers can now write software to enable their software or devices to communicate openly with the OEM's printer.

Carey Lening discusses the issue of reverse engineering: "…The desire to reverse engineer, be it for the good of mankind, the good of profit, or the good of making the software better seems lost amidst the act. The very notion of reverse engineering is given a nefarious tone, no matter what the motivators…" When you own the source you're not reversing anything. You're looking at the code, and making modifications as such. There is no illegality because the source is accessible to all for that reason…" (However) "When you 'own' the binary bits, you have no such freedom."

Open source proponents seek to grow their base by achieving interoperability with proprietary products, many admitting that achieving this without the source code is a tenuous and difficult topic. Unfortunately, most reverse engineering of proprietary software by the open source community tends to be done in adverse situations. If original owners do not release their source code, open source programmers end up working non-cooperatively with the owners to make products interoperable. Nat Ersoz an electrical engineer comments, …If on the other hand, you do not have access to source code or specifications, and would like to figure out how to make two systems interoperate, then that might involve reverse engineering…I say that reverse engineering is typically a hostile and frustrating undertaking…I say that reverse engineering is a "hostile" action, because cooperation between two entities is, for one reason or another, disallowed."

Commenting on copyright law and reverse engineering, Noel Humphries, senior counsel at Akin Gump's licensing practice in New York explains, "The source code that the software writer writes (by making it up out of his head) is covered by copyright by virtue of its being typed into the computer. Regardless of what happens to it later, that bit of code is protected by copyright law from unauthorized copying. Think of a book. The particular sequence of the book's words are protected, whether the book ever gets published or not. The point of a license is to set the terms by which someone can possess a copy of the software, while the copyright holder retains "ownership." In that circumstance, ownership is limited by licenses granted to others. "Ownership" may be attributed to someone, but someone else has legal rights to possess a copy and do something with it. For example, a company may come into possession of software that allows the user to see the code. However, the legal right to see the code does not necessarily create legal authority to modify the code or create a wrapper around it. You would have to look at the terms on which the firm came into possession of the code to determine what the firm was legally empowered to do with that code."

The techniques, processes, and tools to write interoperable open source code will inevitably intersect intellectual property law in the courts. IP infringement has staggering economic implications. If software is freely re-engineered, it will inevitably impact the value of software on the market. If the price of software is adversely impacted, salaries and inevitably employment of programmers will be as well. It can be expected that reverse engineering will be challenged more aggressively in the years to come. Humphries notes, "Copyright law covers the source code. In fact, the human-readable code is exactly what copyright law covers, although code compiled in machine-readable form may be a derivative work of the human-readable code. Copyright is supposed to encourage publication. Copyright law forbids unauthorized copies. Viewing source code is not typically a violation. Taking inspiration from it is not typically a violation. Copying without authority is."

As backlash against reverse engineering, proprietary software companies circumvent open source and GPL licenses, further building distrust amongst both parties. There are a number of approaches that are commonly used to bypass GPL license restrictions. Companies choose to work directly with the author of a product to produce a second product (similar to the first) under a different license. Another strategy is to reverse engineer a GPL product in a different programming language, that way, there would exist two completely different sets of source code for the same product. Another strategy is a technique called dynamic linking. This is best described as using GPL code to create a new product, but the derived (new) product's architecture is such that it is only linked to the GPL software. Files in the two software applications interact and enable the two products to work together. The derived work and the original GPL work can exist under two separate licenses. However, the new software will only work in combination with the previous product, not separately.

Open source code is not guaranteed nor does it come with a warranty. Open source products are often distributed without manuals, instructions or technical information. While a commercial developer is obligated to produce manuals, diagrams and information detailing the functionality of their products, open source programmers are not. In addition, open source developers cannot be expected to create software manuals with the vigor of private firms that are obligated to produce them. Producing technical specifications (in soft or hard copy format) is time-intensive and expensive. But this is not just a customer service issue.

GPL open source or "copyleft" reverses the intellectual property model. For example, in the real world, it would be absurd for users of a technical manual to have permission to modify it in any way and republish it as theirs. More importantly, when a manuscript is bought, because of the enforceability of copyright law, we know that it has not been altered in any way, thus the copyright ensures a buyer (or user) of the manual that he has a genuine copy of the 'Real McCoy'.

Innumerable questions surround the distribution of technical information in the copyleft environment. Issues include: Who should have the right to alter software manuals? Who is the final editor or is there one? How should changes be regulated? Are manuals copyright protected documents? What is the process for making changes? What body regulates these changes? How can organizations guarantee that information in manuals is always accurate?

On a lighter note, while many open source enthusiasts are proponents for copyleft, they insist on copyright and trademark protection for their ideas. The Free Software Foundation, the Open Source Initiative and a number of other organized GPL enthusiasts protect their "marks" by posting notices in publications and on their websites that their trademarks are protected. [2]Recently, it was reported by ZD Net reporter Matthew Broesma reported that that even Red Hat was seeking patent protection as a defensive measure. If you read the OSI website there is a conspicuous notice which reads, "… To identify your software distribution as OSI Certified, you must attach one of the following two notices, unmodified, to the software, as described below. *"This software is OSI Certified Open Source Software…OSI Certified is a certification mark of the Open Source Initiative…"* The same is true for a number of prominent open source firms including VA Linux. While each of these firms would insist that they are not against copyright protection, invoking the protections argues that they certainly appreciate the value in having protecting the ownership of words and/or ideas as valuable real estate.

<u>VI. The Economics of the GPL</u>

When a software product is sold, it represents the efforts of a diverse team of individuals. The revenue from software compensates engineers, graphic artists, database programmers,

[2] Red Hat Defends Patent Applications, Matthew Broersma, ZD Net, June 3, 2002

hardware specialists, debuggers and a multitude of contractors, partners and vendors. In the U.S., the software sector accounted for approximately 319 million jobs in 2001
(see Appendix 8). Software development usually reflects very thin operating budgets and small margins for mistakes. Even after a software application is released, it is often not profitable until its second or third version. The developer must finance both the initial development phase and later modifications. Modifications include: bugs, end-user change requests or upgrades. In the end, successful software is the result of countless hours of programming and extensive capital outlays.

Open source principally perpetuates itself because there is an avid pool of experts and enthusiasts willing to spend their spare time to provide "fixes" and modifications to open-source software. This volunteer system works well as an academic model, but is questionable as business one. It would be tenuous at best, to expect a volunteer model to compete with a for-profit model. Karl O. Pinc, President of the Meme Factory, a software design and consulting firms comments, "Open source software has a reputation for providing rapid fixes. This is true, but probably exaggerated. On the pros and cons of using fixes and patches from the open source community on the Internet, Pinc comments, "Consistency of response most distinguishe fixes for which you pay, as compared to fixes which you obtain for free…Another difficulty is simply in the medium of communication, which is usually text via e-mail or bulletin board. The delay, time, and effort involved in reading and writing reduces the effectiveness of the communication and can lead to misunderstanding. FAQ may be poorly written or mis-read. These problems are eliminated or reduced when obtaining paid support."

The underlying premise of capitalism is that compensation is the best universal tool for motivating individuals to succeed. Full-time programming teams produce innovations for pay and turn to IP protection to market their products. However, without an incentive to create commercial software, filings for copyrights and patents would immediately decline. Thus, it can be expected that innovation would be adversely impacted if the financial incentives for innovating were affected.

The underpinning of the GPL is the argument that if you remove the existence of proprietary software, you will subsequently increase the value of a programmer's services. Programmers would be paid in an open source world, but all software will be non-proprietary and programmers will still be needed to adapt and modify products that will lead to a demand for programmers, etc. David Evans, senior vice president of NERA commenting on the tenuous nature of this dilemma wrote, "The thousands of programmers around the world who volunteer their time for open-source projects provide the movement's greatest strengths. But one wonders how much effort these volunteers will offer to projects that are important to the future success of open source software but that are neither intellectually challenging nor of immediate utility to themselves." Echoing this point on a lighter note, Leo Cooper, open source developer comments, the major disadvantage of the GPL is that it does not seem to create programmer millionaires…"

Another consideration is that support costs over the duration of a software application's availability, in some cases inevitably become the largest cost to the software developer. Customer service costs are harder to manage because consumers are rarely willing to pay for customer service, especially when they are buying low-cost software.

Proprietary development firms must guarantee delivery, operation and maintenance of products. Conversely, even if a firm had the best open source team on their payroll, open source inherently offers no guarantee or warranty whether it comes with a virus, backdoor or any other serious technical problem; and GPL open source only exacerbates these shortcomings.

A remaining consideration is that if open source is successful, it will return goods, services and savings to consumers. Consumers in turn will be able to buy open source applications for home appliances, etc. However, this theory is predicated upon consumer interest in open source products. Discussing the consumer demand of open source, Andre Carter, President of Irimi Corporation a technology consulting firm in Washington comments, “The question of open source code is about whether the software developer wants to make available to the world the blueprint of what they built or simply the benefits of what they built. The notion of open source software has nothing to do with “free software”. The purchase price of computer software is only a fraction of the total cost of ownership. So even if the price tag reads “free”, it can end up being more expensive than software you buy. This is especially true for the typical consumer. If it requires technical know-how to operate, doesn’t offer built-in support, and demands constant attention, it won’t feel free for very long.”

Consumer demand is a very real issue. Evans comments, “There does not appear to be any method for the open source software method to identify and respond to the needs of nontechies…Moreover, because open-source software tends to be written for use by IT professionals, there are few incentives to smooth out the rough edges and make software easier to use. And that limits the prospects for open source in the software used by vast majority of consumers.” Carter continues, “The notion of free software has to survive in the marketplace. Its in the marketplace that we’ll discover if free software can not only address a specific defined need, but also countless undefined, unconscious needs. Those are the needs the consumer doesn’t even know they have. That’s a software product, a total solution. Market forces will decide if it offers that total solution to consumers. There are all types of consumers with ranges of needs and abilities. The guys in the lab at MIT don’t need install wizards, plug and play drivers, voice based technical support and big picture manuals as part of their software. However, the elderly couple e-mailing their grandkids or the mother of two managing accounts on a PC in the kitchen does. With most free software when you have a problem, you may have access to some online forum to discuss that problem in a chat room with tech types, but is that of interest to the average consumer? But the problem is yours to fix, that’s beyond most consumers. In the end, the free software could end up awkward and costly to the consumer. This point cannot be ignored.”

Proportionality in GPL exchanges becomes an important economic consideration. If a software application representing 5000 hours uses GPL code that reflects only 100 hours, is the GPL fair in its argument that the entire product is GPL? This point is of considerable concern to software companies that value their secrets, design and architecture strategies. Proponents of the GPL argue that each party in the exchange is benefiting equally, but without a means to properly make this evaluation, this position at best is over-assuming.

The significant investment of time and resources to develop software explains why developers with the hopes of recouping their investment would be unwilling to give away something that reflects years of work and investment. The developer must recoup research and development costs as well as make a profit. This uphill challenge discourages interest in the GPL, a license that stipulates that any use of GPL code mandates "fair" exchange of source code.

Hundreds of firms would have to agree on making considerable investment in open source systems, training and upgrades to their existing software. Thus, the switch would have to offer substantial and immediate cost savings, and a significant comparative advantage to staying with the status quo. In addition, customers are more speculative today. The market and the credibility of the tech sector has made companies hesitant to buy tech solutions that have not been tested thoroughly. Investments based on long-term profit assumptions are particularly unpopular due to the collapse of many of the .coms.

The best ideas in technology sit on the shelf for decades. Their adoption is almost always gradual, particularly because people have to agree that the new system offers significant advantages over the old way of doing things. Regardless of the environment, people are resistant to change. While some .com firms did have technology and delivery problems, most failed because they needed a massive adoption of their ideas and concepts for shopping to occur at once. For all programmers at once to decide that all software should be free and the only thing of value would be their services is a leap in practicality and reality. The classic allegory of the prisoner's dilemma suggests that in fact programmers would be slower, particularly if they suspected their colleagues had conflicting intentions. Even in a vacuum of trust, each programmer, organization and company would be waiting for the other to forgo their current model of collecting profits to make their software open source.

The success of an A-Z open source environment would expectedly impact the software sector as a viable entity. If software is freely available, but PC's, servers and hardware maintain their value, we can only predict that the value of software companies will plummet. Hardware will come with more and more free software. Second, we can only expect that the revenues and value of the software sector will transfer to the hardware sector. Although the software sector has seen growth almost every year, it is questionable whether the GPL model will enable the software industry to continue its exceptional growth particularly when the growth in the software sector is tied to proprietary products, something the GPL is anxious to eliminate.

GPL presents a host of financial questions to corporations. While one company may decide to contribute its source code, they have no guarantee that a competitor is doing the same. Another consideration is that prevailing doubt concerning the GPL has created an environment that is actually encouraging misuse of GPL source code. If companies are concerned about any variation of these issues, it subsequently affects the value of the GPL license as well as its fair distribution of source code.

Businesses must be concerned about the perception of the GPL. For example, experts assess the value of intellectual property when completing valuations of firms. Because GPL open source literally erases the proprietary and trade secret value of software, it can be expected that firms concerned about valuations will be very concerned about using GPL open source. This is very serious when considering that mergers, acquisitions, initial public offerings (IPO), and loans are just a few examples of critical transactions tied to the valuation a company. Even small firms that are avid users of open source to move the development of their projects fast forward, might be shunned by investors or other third parties because of their use of GPL products.

VII. GPL Open Source and the Courts

The acquisitiveness of the GPL inherently creates legal issues for all parties involved. However, without legal precedent to offer definitive answers, many companies are on the sidelines awaiting the courts to provide a legal interpretation of the General Public License.

Once GPL code is combined with another type of source code, the entire product is GPL. Subsequently, this change could occur deliberately, but it could also occur accidentally. There are unlimited scenarios for accidents to occur, the license could be lost in the source code's distribution, or maybe unreadable due to a glitch in its electronic distribution. Another potentially litigious issue is whether the use of GPL tools used to manipulate code subject software to the GPL. Theoretically, a GPL tool could subject new software to GPL restrictions. This too will have to be interpreted by a judge. Regardless, unknowing users of GPL might have one intention for use of the license and find out later that it inadvertently infringed upon copyright protected work. Legal questions relevant to such an event intersect the legal arenas of intellectual property rights, contract law and liability.

Another issue is the question of author rights. If the author decides that you are not distributing the program in a manor that meets his satisfaction, he could decide to withdraw your right to distribute the program. Again, this awaits a decision by the courts. It is not clear how the GPL exposes a licensee to this restriction. GPL licensees also will need clarification on this issue if they sub-license the product. Again IP, contract law and liability questions are irrelevant if a licensee violates the GPL.

Section 2 of the GPL says that identifiable sections of a work that are not derived from a GPL program and that "can be reasonably considered independent and separate" are not subject to the GPL when distributed as separate works. This becomes a very subjective issue as well. It is questionable whether an owner could accurately identify a work as under the original GPL code or otherwise.

Section 2 also says that a "mere aggregation of another work not based on the [GPL] Program on a volume of a storage or distribution medium does not bring the other work under the scope of this License." This issue also leads to legal questions, forcing licensees to have to decide (to the best of their ability) whether combined products work as a GPL or non-GPL product. Use of GPL code that includes code licensed from a third party could obligate you to sublicense the other party's code under the GPL. In other words, if you buy a bucket of apples under a restrictive agreement that happens to include a couple oranges that you like, it is questionable what rights you have to buy (or sell) oranges you get directly from the supplier once the previous agreement is in place. Likewise, it is questionable if the owner of code that is GPL'ed can change his mind and distribute it with a new license. Each of these scenarios questions the ability of GPL licenses to follow source code that becomes GPL. Inevitable jurisprudence will tackle these and other gray areas of the GPL.

VIII. Conclusion

Open source as a development model is helpful to the software industry. For example, software distributed under the BSD license is very popular. The BSD license (see Appendix 9) enables companies, independent developers and the academic community to fluidly exchange software source code. In addition, developers frequently create their own versions of open source licenses to independently distribute their source code.   Joseph Baker, a systems administrator in Wisconsin comments, “GPL software is a different horse though. Proprietary vendors could use some GPL software to make their own networks more reliable, easier to maintain and less expensive to equip their employees to get the job done. But proprietary developers cannot integrate GPL software into their binary files without then changing their own software license to the GPL. …It’s interesting to note that while BSD projects can be integrated into GPL ones, GPL projects cannot be distributed under a BSD style license.”

The concerns about the GPL range in opinion. Steve Wilson a computer designer comments, “I’m not sure either license is appropriate. I much prefer original method, i.e. public domain. Perhaps that equates in most case to the BSD license, but public dollars paid for it, then all of the public should have the right to take advantage of it. If I do a value-add to this software, then I believe it should be my choice to decide what license I stick on the code I’ve added.”

The GPL’s resistance to commonplace exchange of open source and proprietary has the potential to negatively impact the research and development budgets of companies.   IBM, Sun and Microsoft alone combine for over ten billion dollars spent annually on research and development. It stands to reason that if the ownership of intellectual property is affected, dollars spent on research and development would be at risk as well. Removing the economic incentive for firms to own the rights to products spawned from research and development programs is the surest way to end their existence.

The GPL has many risks, but the greatest is its threat to the cooperation between different parties who collaborate and create new technologies. Today, government, commercial enterprise, academicians, etc. have a system to converge. Conversely, the GPL represents divergence; proposing to remove the current infrastructure of intellectual property rights, enforceable protection and economic incentive.

Widespread adoption of GPL should be carefully surmised. While there is an activist nature alive encouraging its use, within these groups are individuals that insist on reporting the truths, not myths about GPL open source. In addition, some feel that the licensing debate has other motives. Peter Sestoft, Associate Professor at the Royal Veterinary and Agricultural University in Denmark comments, “The BSD licensing agreement permits the subsequent use of open source in closed source software projects; the GPL does not. Choosing one over the other is mostly a political issue. I don’t object to either of them.”

[3]”Lori MacVittie of Network Computing described Sun’s ambivalence towards the GPL writing, “Don’t let Sun Microsystems’ recent overture to the open-source community fool you…Rather than jump into bed with the open-source community, Sun is keeping one foot firmly on the ground. It refuses to accept any API issued under what it considers “viral” licensing – the General Public License in particular – and will continue to maintain complete control over the Java language specification.”

[3] Sun’s Tepid Romance With Open Source Will Benefit Mobile Apps, Network Computing, April 29, 2002

While GPL advocates are quite active in their promotion of copyleft, few would disagree that its widespread adoption would present a radical change to an industry sector which generates close to 350 billion dollars in sales annually worldwide (see Appendix 10). Open source can coexist with the status quo, but GPL cannot coexist with traditional open source or proprietary source code. Considering the varied implications of the use of GPL open source, its continued use clearly deserves further study.

Important considerations include:

1- Engineering software has become considerably complicated and rigorous. It is not unusual for software to include millions of lines of source code. If the incentive to develop software is changed, we can subsequently expect the quality and efficiency of software to change.

2- There remain considerable differences within the GPL open source community. It is questionable whether these groups will continue to be proponents of the GPL in its current form or opt for changes in the immediate future.

3- Open source has successfully been used in proprietary software. In addition, academic and government projects have been successful particularly because of commercial interest. Private enterprise offers unique efficiencies for the success of government funded research.

4- Open source GPL use by government agencies could easily become a national security concern. Government use of software in the public domain is exceptionally risky.

5- Reverse engineering, perpetuated by GPL proponents, threatens not only the owners of intellectual property, but also the software industry itself.

6- Use of GPL open source creates a number of economic concerns. For example, the valuation of a software company could be significantly affected if it uses source code licensed under the GPL for the development of its products.

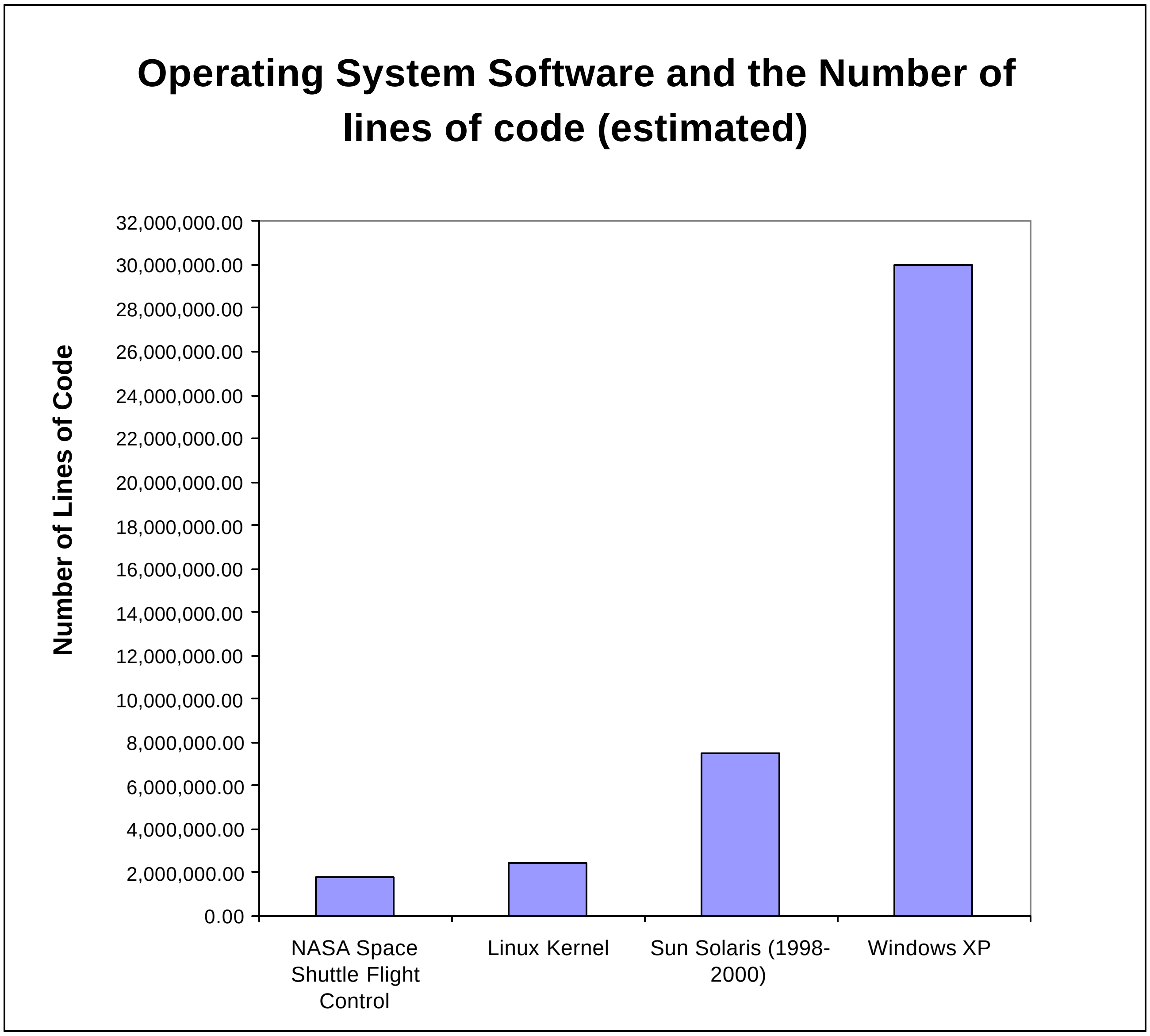
7- The courts have yet to weigh in on the General Public License. Without legal interpretation, the use of the GPL could be perilous to users in a number of scenarios.

8- For each advantage adoption of the GPL presents, there are considerable tradeoffs which mandate further study of the issue.

IX. Bibliography

Open-Source Debate Rages, Ashlee Vance, Network World, 07/27/01
Open-Source Debate Rages at O'Reilly Conference, Ashlee Vance, IDG news service, 07/26/01
Storming the Gates, Nathan Newman, The American Prospect Online, 3/27-4/10/2000
Open Source Versus Free Software Debate Rages, Dietmarr Mueller, 06/25/01
Science versus Capitalism: The Open Source Debate, Niki Scevak, Internet News, 5/15/01
Are Open Source and Innovation Compatible? Niall Murphy, Embedded.com, 5/8/02
Open Source is Already Delivering, Bill Gatliff, Embedded.com, 5/8/02
Linux as an Embedded Operating System, Jerry Epplin, Embedded.com, 10/97
Can Linux Billionaires Carry the Free Software Torch? Andrew Leonard, 12/23/99
Working Without Copyleft, Bjorn Reese and Daniel Stenberg, O’Reilly Network, 12/19/01
Open Source, Standards and Windows, Larry Seltzer, ZD Net, 1/22/02
More Than a Gigabuck, Estimating GNU/Linux’s Size, David A. Wheeler, 6/01
Red Hat Defends Patent Applications, Matthew Broersma, ZD Net, June 3, 2002
Sun’s Tepid Romance W/Open Source Will Benefit Mobile Apps, Network Computing, 4/29/02

Appendix I – Comparative Code Chart[4]



[4] David A. Wheeler: "More Than a Gigabuck, Estimating GNU/Linux's Size". www.dwheeler.com

Appendix II – Software Publishers: 1998 - 2000 (Millions of Dollars)[5]

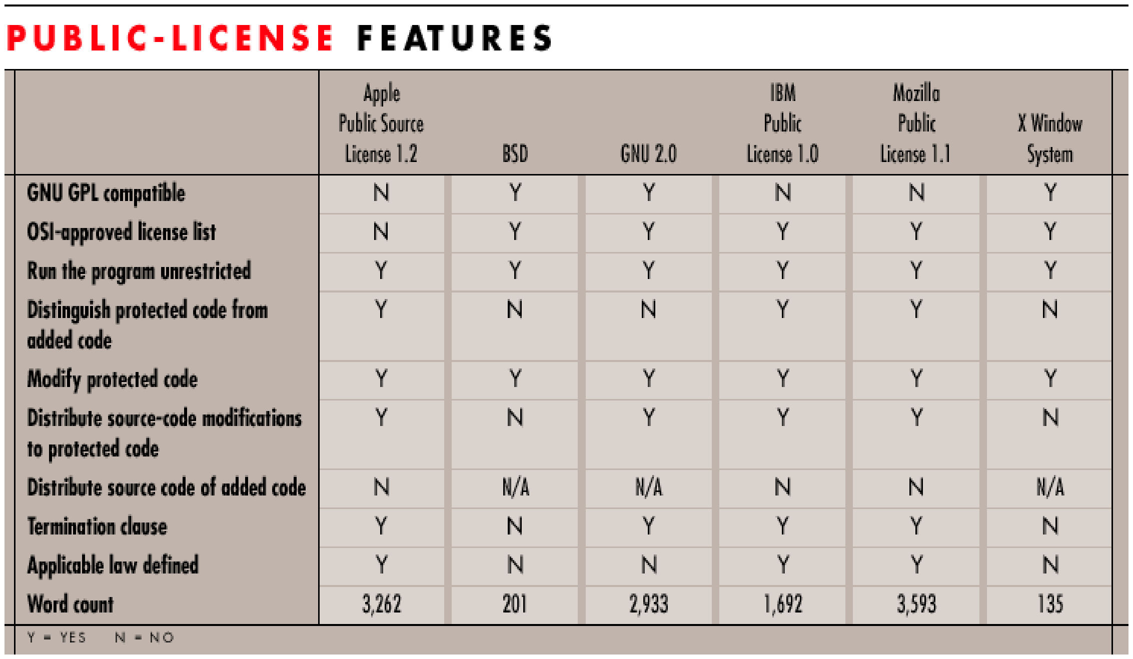
SOURCES OF REVENUE	2000	1999	1998	2000/ 1999	1999/ 1998
Personal computer software revenue, total	$13,819	$12,940	$11,818	6.8%	9.5%
Consumer applications	7,637	7,974	7,247	–4.2	10.0
Vertical market applications	2,643	2,308	2,324	14.5	–0.7
Software development tools, languages, and utilities	S	1,254	1,095	S	14.5
Enterprise software revenue, total	$ 24,755	$22,849	$20,580	8.3%	11.0%
Cross-industry applications	16,084	15,003	13,194	7.2	13.7
Vertical market applications	5,808	5,532	5,426	5.0	2.0
Software development tools, languages, and utilities	2,862	2,313	1,960	23.7	18.0
Systems and systems management software revenue	$13,809	$12,261	$10,130	12.6%	21.0%
Mainframe computer software revenue, total	$ 8,700	$8,676	$8,065	0.3%	7.6%
Operating system software	4,749	4,854	4,669	–2.2	4.0
Database software	2,856	2,699	2,483	5.8	8.7
Networking software	S	1,122	913	S	22.9
Services revenue, total	17,443	16,076	13,625	8.5	18.0
Implementation and customization	5,727	5,428	4,597	5.5	18.1
Software upgrades and maintenance	9,431	8,575	7,224	10.0	18.7
Software user training	1,580	1,313	1,201	20.3	9.3
Web site advertising	49	35	15	39.9	138.5
Other revenues	6,800	6,883	7,117	–1.2	–3.3

[5] SOURCE: Estimates are based on data from the 2000 Service Annual Survey and administrative data. Links to this information on the Internet may be found at www.census.gov/svsd/www/cv.html. S data do meet publication standards because of high sampling variability or poor response quality. Some unpublished estimates can be derived from this table by subtracting published data from their respective totals. However, the figures obtained by such subtraction are subject to these same limitations. These unpublished data are for internal use only.

<u>Appendix III</u> – <u>Short Sample List of Open Source Licenses</u>

The GNU General Public License (GPL)
The GNU Library or "Lesser" Public License (LGPL)
The BSD license
The MIT license
The Artistic license
The Mozilla Public License v. 1.0 (MPL)
The Qt Public License (QPL)
The IBM Public License
The MITRE Collaborative Virtual Workspace License (CVW License)
The Ricoh Source Code Public License
The Python license (CNRI Python License)
The Python Software Foundation License
The zlib/libpng license
The Apache Software License
The Vovida Software License v. 1.0
The Sun Industry Standards Source License (SISSL)
The Intel Open Source License
The Mozilla Public License 1.1 (MPL 1.1)
The Jabber Open Source License
The Nokia Open Source License
The Sleepycat License
The Nethack General Public License
The Common Public License
The Apple Public Source License
The X.Net License
The Sun Public License
The Eiffel Forum License
The W3C License
The Motosoto License
The Open Group Test Suite License

Appendix IV – Open Source License Variations[6]



PUBLIC-LICENSE FEATURES

	Apple Public Source License 1.2	BSD	GNU 2.0	IBM Public License 1.0	Mozilla Public License 1.1	X Window System
GNU GPL compatible	N	Y	Y	N	N	Y
OSI-approved license list	N	Y	Y	Y	Y	Y
Run the program unrestricted	Y	Y	Y	Y	Y	Y
Distinguish protected code from added code	Y	N	N	Y	Y	N
Modify protected code	Y	Y	Y	Y	Y	Y
Distribute source-code modifications to protected code	Y	N	Y	Y	Y	N
Distribute source code of added code	N	N/A	N/A	N	N	N/A
Termination clause	Y	N	Y	Y	Y	N
Applicable law defined	Y	N	N	Y	Y	N
Word count	3,262	201	2,933	1,692	3,593	135

Y = YES N = NO

[6] Chart reprinted by permission of Sean Doherty, Technology Editor, Network Computing

Appendix V – The MIT License[7]

[7] – http://www.opensource.org/license/mit-license.html

Appendix VI – The GNU Public License (www.gnu.org)

Appendix VI – The GNU Public License (www.gnu.org)- continued

that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you". Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

Appendix VI – The GNU Public License (www.gnu.org)- continued

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the sc ripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by

Appendix VII – Total Counts by License[8]

```
 15185987 (50.36%) GPL
  2498084 (8.28%)  MIT
  2305001 (7.64%)  LGPL
  2065224 (6.85%)  MPL
  1826601 (6.06%)  Distributable
  1315348 (4.36%)  BSD
   907867 (3.01%)  BSD-like
   766859 (2.54%)  Freely distributable
   692561 (2.30%)  Free
   455980 (1.51%)  GPL, LGPL
   323730 (1.07%)  GPL/MIT
   321123 (1.07%)  Artistic or GPL
   191379 (0.63%)  PHP
   173161 (0.57%)  Apache-like
   161451 (0.54%)  OpenLDAP
   146647 (0.49%)  LGPL/GPL
   103439 (0.34%)  GPL (programs), relaxed LGPL (libraries),
                   and public domain (docs)
   103291 (0.34%)  Apache
    73650 (0.24%)  W3C
    73356 (0.24%)  IBM Public License
    66554 (0.22%)  University of Washington's Free-Fork License
    59354 (0.20%)  Public domain
    39828 (0.13%)  GPL and Artistic
    31019 (0.10%)  GPL or BSD
    25944 (0.09%)  GPL/BSD
    20740 (0.07%)  Not listed
    20722 (0.07%)  MIT-like
    18353 (0.06%)  GPL/LGPL
    12987 (0.04%)  Distributable - most of it GPL
     8031 (0.03%)  Python
     6234 (0.02%)  GPL/distributable
     4894 (0.02%)  Freely redistributable
     1977 (0.01%)  Artistic
     1941 (0.01%)  GPL (not Firmware)
      606 (0.00%)  Proprietary
```

These can be grouped by totalling up SLOC for licenses containing certain key phrases:

```
 16673212 (55.30%) GPL
  3029420 (10.05%) LGPL
  2842536 (9.43%)  MIT
  2612681 (8.67%)  distributable
  2280178 (7.56%)  BSD
  2065224 (6.85%)  MPL
   162793 (0.54%)  public domain
```

[8] David A. Wheeler: "More Than a Gigabuck, Estimating GNU/Linux's Size". www.dwheeler.com

Appendix VIII – Jobs Created by Software Industry (Millions)[9]

Packaged Software Employment

1991-2001

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Annual
1991	119.9	120.6	121.3	121.4	122.7	125.2	126.5	126.3	126.5	126.7	127.3	128.6	124.4
1992	128.6	127.9	128.8	129	129.5	131.7	133	131.8	130.7	131.9	132.8	134.1	130.8
1993	138.2	139.1	140.1	140.9	142.3	144.2	146.8	148.1	148	149.3	149.7	150.9	144.8
1994	152.4	153.1	153	152.6	153.1	154.9	157.6	158.4	161.7	161.5	163.7	166.2	157.4
1995	167.7	170.2	173.3	175.4	176.7	180.6	183.2	185.4	187.4	187.1	189.8	192.5	180.8
1996	192.8	194.3	195	195.5	195.9	197.4	200.9	204.1	205.4	207.8	210.4	212.4	201
1997	214.2	217.3	219.4	221.1	221.4	225	226.4	226.1	228.8	230.4	230.6	233.6	224.5
1998	236	238.6	241.4	243.8	244.6	248.1	253	251.9	250.7	250.9	251.8	253.1	247
1999	258.5	262.6	264	265.8	267.5	270	273.6	274.1	275.3	274.6	276.7	278.8	270.1
2000	280.2	282.6	287.4	291.7	295.2	301.1	307.2	310.5	310	309.4	310.5	313.2	299.9
2001	316.5	319	321.3	320.5	317.9	322.2	320.8	321.3	321.3	321.4	319.4	317.2(p)	319.9(p)

[9] Bureau of Labor Statistics, Current Employment Statistics, http://www.bls.gov/ces/home.htm#data. p : preliminary

Appendix IX – The BSD License[10]

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

[10] www.opensource.org/license/bsd-license.html

Appendix X – Software Sales Worldwide: 1996 – 2001 (In Millions)[11]

	1995	1996	1997	1998	1999	2000	2001*
Total Software	$ 92,000	$ 105,000	$ 122,000	$ 140,300	$ 156,848	$ 171,146	$ 182,784
US	$ 41,400	$ 47,250	$ 54,900	$ 63,000	$ 81,413	$ 90,579	$ 96,738
W Europe	$ 30,820	$ 35,175	$ 40,870	$ 46,000	$ 47,166	$ 49,008	$ 52,721
Asia/Pacific	$ 7,728	$ 13,440	$ 16,104	$ 18,091	$ 19,290	$ 21,676	$ 22,933
App Development Tools	$ 26,100	$ 27,500	$ 31,000	$ 33,500	$ 36,500	$ 45,419	$ 47,554
Applications	$ 39,900	$ 46,000	$ 54,000	$ 63,000	$ 72,000	$ 80,175	$ 86,500
System Software	$ 26,400	$ 30,800	$ 35,550	$ 41,000	$ 47,800	$ 48,125	$ 50,339
Growth Rates							
		1996	1997	1998	1999	2000	2001
Total Software		13.8%	16.2%	15.0%	11.8%	9.1%	6.8%
US		14.1%	16.2%	14.8%	29.2%	11.3%	6.8%
W Europe		14.1%	16.2%	12.6%	2.5%	3.9%	7.6%
Asia/Pacific		73.9%	19.8%	12.3%	6.6%	12.4%	5.8%
App Development Tools		5.4%	12.7%	8.1%	9.0%	24.4%	4.7%
Applications		15.3%	17.4%	16.7%	14.3%	11.4%	7.9%
System Software		16.7%	15.4%	15.3%	16.6%	0.7%	4.6%

[11] IDC Gray Sheet Revised 12/01. *Revised 11/26/01 for 9/11 impact. "Application development tools" include programming tools, programming languages, RDBMS like Oracle. "Applications" include client/server enterprise applications, desktop applications, e-commerce and Internet applications.