Legal and Business Assessments of the GPL

Robert H. Lee
Advisor: Robert Dunne

Dept. of Computer Science
Yale University

April 20, 2005
Abstract

The open source model of software distribution has existed since the creation of Unix in 1969. Public licenses, which create legally-binding contractual relationships between a user and developer, enable individuals to copy, modify, and redistribute works of open source software. The GNU General Public License (GPL) is the most popular license used for open source software. Although the GPL should be considered a valid and enforceable contract, a court could strike it down for containing vague key terms. It potentially provides greater benefits to the licensor and the licensee than a proprietary license like Microsoft’s End-User License Agreement (EULA). To maintain a strong position in the IT industry, open source software vendors must emphasize the customer’s rights to copy, modify, and redistribute their products to distinguish their products from those of proprietary software companies. MySQL AB’s dual licensing system should be adopted by other open source vendors because it enables proprietary software developers to create derivatives from open source software without imposing on them the legal obligation to disclose source code.
Table of Contents

1 Introduction 4
2 History of Open Source 16
3 Public Licenses 23
4 Business Aspects 45
5 Conclusion and Further Research 57
Bibliography 59
Appendix: Licenses 61
1 Introduction

Since the 1960s, the open source model of software distribution has been used to develop some of the most successful software in the IT industry, such as the Unix operating system and the Mozilla Firefox web browser. Its products and processes are popular within universities, government agencies, research laboratories, and corporations, and a large and vibrant community of programmers advocates its use. A programmer who wants to contribute to an open source project will find many websites that will afford him the opportunity to implement almost any conceivable software idea. If he does not find a project he likes, he can start his own open source project and invite other programmers to contribute code. The open source model allows a development team to harness more manpower than the proprietary framework. It creates benefits for the entire programming community.

It has had and continues to have a tremendous impact on the technological world and has been extensively studied from many viewpoints. In this paper, we will limit our analysis to the legal and business issues of the world’s most popular open source license, the GNU General
Public License (GPL). There are two main ideas proposed by this paper. The first one suggests that the GPL potentially provides greater benefits to the licensor and the licensee. The typical GPL licensor spends less on legal costs than his proprietary software counterpart because the licensor has fewer conditions to enforce on the licensee and because he loses very little even if these conditions are violated. As for the licensee, the GPL grants him the rights to copy, modify, and redistribute software, whereas proprietary software licenses do not grant any of these empowering rights.

The second idea suggests that open source vendors must emphasize the GPL rights of their products to maintain a strong position in the IT industry. Open source products currently tend to have lower acquisition costs than proprietary software, a quality that benefits vendors who develop and sell such products. However, in time, proprietary software companies will likely find ways to reduce the costs of their products. When they do, companies that sell open source software will have to find a way to distinguish themselves from their proprietary software counterparts. Since proprietary software companies withhold source code from their users, open source vendors can differentiate their products by convincing customers that the rights to copy, modify, and redistribute software are worth the associated costs and risks.

The first section of this paper establishes the definitions of key open source terms to clarify any misconceptions about the open source model of software distribution. It also briefly explores two reasons why developers use the open source model. The second section provides a history of open source. In the third section, we discuss the legal aspects of the GPL and compare it to the BSD license and Microsoft’s End-User License Agreement (EULA). We present two business case studies in the fourth section and conclude with further research questions in the fifth section.
1.1 Definitions

In order to proceed with our discussion, it is necessary to establish the definitions of several terms.

*Closed software distribution* is the distribution of software with a license that prohibits a user from copying, modifying, and redistributing the software. In order to preclude users from modifying their software, closed software developers withhold the source code files that were used to generate their software. To *close up* software means to change a piece of software’s license terms so that users cannot copy, modify, or redistribute it. Closed software can also be called *proprietary software*.

*Open source software distribution* is the distribution of software and its source code with a license that gives a user the rights to copy, modify, and redistribute the software, its source code, and any derivatives thereof. For a piece of software to qualify as open source, its distribution must include all these three parts: software, source code, and rights. Given the numerous means to communicate data to a user, an open source developer can distribute these three parts in two ways. One, if he opts to use optical storage media such as CD-ROMs or DVDs to provide the software, he can also provide on the disks the source code files and a text file enumerating the rights. On the other hand, if he chooses to release his software via a network such as the Internet, he can upload all the source code files and rights file onto a FTP site that would have the software. It is even possible for a developer to combine these two means by providing only the software and rights on an optical storage medium and giving access to the source code via a FTP site. In this case, the user would be able to use the software simply by reading the medium on a disk drive, but she would need a network connection to access the FTP site that contains the source code. If these three items—software, source code, and rights—are included in a distribution, then that distribution can legitimately be called open source.
There is a popular misconception that open source software should also be free of charge to the user. As we will soon discover in our discussion concerning public licenses, this attribute is not required for a piece of software to be open source. If a developer chooses to release his software to a user for a cost but also provides access to the source code and the aforementioned rights, he is still using the open source model of distribution. Additionally, open source software available through purchase does not have to be inexpensive; in practice, if developers choose to fix a monetary price on their software, most of them charge only enough to recoup production costs. However, it is possible under the definition of open source software to charge any amount and still have the software retain its label.

If a piece of open source software can have a price affixed to it, then why do some users think that they should be able to obtain it without paying for it? To answer this question, we introduce the term free software. Free software essentially has the same attributes as open source software and as a result, the two categories can be used interchangeably. The word “free” in free software, however, does not refer to “free of charge,” but rather to “free to copy, modify, and redistribute.” The freedom that free software contains is given to a user via the rights. Since “free” can have meanings referring to either “without charge” or “freedom,” users sometimes mistaken free software as not having a price. Humorously put, the Free Software Foundation (FSF) advises that one should think of “‘free’ as in ‘free speech,’ not as in ‘free beer.’”  

Although free and open source software are very similar in that they both contain in their distribution the three elements of software, source code, and rights, these two categories are nonetheless different because free software licenses are more restrictive than open source

---

licenses when it comes to closing up derivatives of the software.\textsuperscript{2} To illustrate the difference, consider an example where a computer hobbyist downloads an email client program from the Internet. In addition to the client, he is given access to the source code and is permitted to copy, modify, and redistribute the client and any derivatives thereof. Suppose he discovers that the client lacks a functionality that he would like to use with the software. Remembering that he has the right to modify the software under the license, he decides to undertake the task of adding the functionality to the original client. After many weeks of coding and debugging, he has his finished product: an augmented version of the client that now fulfills the original specifications and provides the new functionality.

Excited, he stops to consider what to do next. Should he release the new client and its source code to the group of programmers who wrote the original client? Or, given the hours of programming (and not to mention the higher quality of the new client), should he refrain from releasing it to the public? He realizes that his modification to the client could enable other hobbyists to add more useful features, which would be a benefit for the members of the community who use the client; however, he predicts that further additions could obsolete his feature. This possible consequence does not bode well with him. He decides to close up the software and sell it to a third-party vendor.

Proponents of free software would claim that under the GNU General Public License (FSF’s main public license), this hypothetical programmer would not have the right to close up the client. FSF and Open Source Initiative (OSI), two of the most influential open source organizations, require derivatives of open source software to carry the same copying, modifying, and redistributing rights as the original software. These organizations intend to preclude

developers from closing up derivatives by imposing this requirement. Other proponents of open source, such as the supporters of the Berkeley Software Distribution (BSD) license, would permit developers to close up software that derive from open source software. Individuals of both sides have debated whether open source licenses should permit developers to close up derivatives, and the current list of various open source licenses indicates that the issue has not been satisfactorily resolved. We will discuss the implications of these licenses’ differences in the second section.

There are other terms used in the software industry that could pose additional confusion. A *shareware* is software whose features are intentionally limited to encourage users to purchase the full version wherein all the features are enabled. If shareware versions of software are available, users can first try the shareware before deciding to purchase the full version. Sharewares are unlike open source software because they usually do not allow users to modify the software. Source code files are rarely distributed with a shareware. If the shareware distribution model were to have any connection to open source model, only the misconception that open source software are gratis could provide the basis for comparison: sharewares are often available for no charge or for a nominal fee. However, as we have explained, pieces of open source software do not inherently have to be gratis. Furthermore, unlike their open source counterparts, shareware developers are at liberty to impose restrictions via the license to prohibit users from copying, modifying, and redistributing the shareware. Since some sharewares are actually the full versions with numerous features disabled, if the license were to permit users to modify the shareware, some users would likely find a way to enable the features, which would in effect make the shareware into a full version. With a full version, the user would have no incentive to purchase a full version from the actual developer and the developer’s mechanism for
encouraging the purchase of the full version would become ineffective. Note that the term *freeware* refers to shareware with all of its features enabled.

*Beta versions* are prereleases of software that are distributed primarily to experienced users and third-party developers for the express purpose of stress-testing the software. Like shareware, beta versions of software often do not have any of the characteristics that define open source software. Although the intent of a developer who releases a beta version is to detect hidden bugs and problematic performance issues, he chooses not to give the third-parties any access to the source code. Thus, given a beta version, the only way for these users to detect problems is to use the software in real-life environments with naturally-occurring input values (as opposed to contrived, artificial values). They cannot help improve the software by examining the source code because they are not given access to the code.

Lastly, as a matter of clarification, when either terms “free software” or “open source software” could be correctly used in a given situation, we choose to use the latter term.

### 1.2 Why Is Open Source Used?

Consider a situation involving a network administrator who is seeking to improve the performance of his organization’s server. He discovers that an open source software will sufficiently fulfill his goals. Unfortunately, he also learns that the software is only available for a considerable sum of money, one which his organization is not willing to pay. As he is about to give up on acquiring the software, the in-house counsel comes into his office and advises him that if he can convince the network administrators of other organizations to ask their managers for resources to pay for part of the software, then together they can acquire the software and make copies of it for every contributing organization. He contacts his colleagues in other organizations and they agree to ask their managers for a small amount of money for the
cooperative purchase of the software. Their managers grant their request and one copy of the software is purchased. Upon receipt of the software, copies are made and distributed to every contributor.

In this scenario, a group of admittedly cunning users together bore the cost of purchasing a piece of open source software and made copies thereof for each user. Undoubtedly, they took advantage of the copying and redistribution clauses in the license of the software: with one purchased copy, they provided each contributor with a copy of the software at a cost much lower than the cost of purchasing it directly from the developer. This technique seems to be bypassing the developer’s mechanism for generating revenue. If the developer knows that this cooperative technique can be used to avoid paying the full price for the software, should he not consider either closing up the software or removing the copying and redistribution clauses to ensure that he receives payment for his software?

Developers of such software would respond to the above question with a “no.” To understand why they would respond this way, consider why a developer would impose a charge in the first place. He would likely impose a charge to recoup the production costs of writing the software and maintaining a website to advertise it. His primary concern in charging money to his users would be to ensure that the costs do not prevent him from pursuing other open source projects in the long-run. If users choose to distribute copies of the software using their own equipment, they are the ones who are incurring the costs to distribute, not the developer. Since the developer does not have to pay for the users’ decision to distribute his software, he is not concerned from a business point-of-view about preventing users from bypassing the requirement to pay.
This lack of concern raises the general question: why do some developers use the open source model of distribution? Furthermore, what motivates them to release the source code of their software? The term “source code” aptly includes the word “source”: from this code, the entire software is generated. Should not a developer withhold such valuable information from users?

There are two reasons why developers use open source. The first one is sociological: the act of sharing source code with a community of users and developers increases the level of software quality available to the community as individuals improve on existing code. Since the community shares its software with all of its members, improvements carried out by a subgroup within the community benefits other members. Proponents of open source software appeal to this reason to justify the open source model. They assume that programmers who have access to source code of useful software will seek to improve the software. Historical events can support the validity of this assumption. The Linux operating system (OS), the epitome of open source software, was produced by “several thousand developers”\(^3\) who sought to enhance its functionality through correctly and cleanly written code. As a result of their effort, Linux is currently one of the most widely used OSs and can be found on corporate servers, government supercomputers, and university workstations. Its popularity testifies to the success of sharing and improving source code within a community.

These proponents assert that software whose source code is hidden to a user will evolve much more slowly than software whose source code is available for public access and modification. As evidence, they inevitably compare Microsoft Windows to Linux and state that Windows has suffered from numerous security attacks, whereas Linux has not. OSI argues that

“open-source operating systems and applications are generally much more security-safe than their closed-source counterparts. When the ‘Ping o’ Death’ exploit was revealed in 1997 […] Linux had fix patches within hours. Closed-source OSs didn’t plug the hole for months.”4 They might also make reference to the code testing company Coverity’s recent finding that the source code for the open source database software MySQL had merely ninety-seven flaws; most proprietary software have many more.5

The nature of debugging and stress-testing software suggests that access to source code will always lead to safer and more robust programs. Seasoned programmers will argue that although stress-testing is a legitimate way of verifying that a piece of software conforms to its specifications, examining its source code constitutes a more efficient and more rigorous means of ensuring that a secure program is eventually released to the public. To understand why the ability to examine source code helps a developer to detect and resolve a program’s problems with greater efficiency and thoroughness, consider the following analogy to a black box. A programmer who is trying to make a piece of software safer without access to its source code is essentially trying to figure out the behavior of a black box without being able to examine its internal workings. Particular inputs will generate particular outputs, and over time, with a large collection of input-output pairs, the black box examiner will make reasonable inferences about the box’s internal construction. However, because the examiner has no way of verifying his inferences, he can never be certain that any of them are correct. Similarly, a tester without access to a piece of software’s source code can only make reasonable yet unverifiable deductions about the software’s behavior given certain input values. Programmers know by experience that

it is almost always an unanticipated input value that leads to program crashes. Even with software that has withstood extremely rigorous stress tests, developers are hesitant to claim that it will be secure and functional in a real-life setting. However, by being able to examine source code, a tester can verify his inferences and prove correctness properties for a piece of software, which would have been impossible without access to the code.

The second reason that proponents of open source use to justify the open source model is related to business: open source is a viable business venture. Bob Young, Chairman and CEO of Red Hat, believes that “[t]he success of any industry is almost directly related to the degree of freedom the suppliers and the customer of that industry enjoy.”6 Economically speaking, when a market has many competing firms, each firm has to differentiate its product from those of its rivals to maintain and increase its market shares. The process of differentiating its products leads to innovation and higher-quality goods, which are benefits to the consumers. However, when a market is controlled by a small number of firms, the monopolists have little incentive to innovate: innovation requires ingenuity and investment of resources, and since they hold large shares of the market, rational managers would forego the decision to improve their goods. Consumers purchase their goods, despite a preference for more innovative goods. In this case, firms are slow to improve their goods because they lack the incentive to do so.

Young cites two examples to support his belief. One, the telephone industry witnessed an unprecedented degree of innovation when the American Telephone and Telegraph Company (AT&T) lost its monopoly in the telephony market. Without AT&T’s control, small and large businesses were able to provide consumers with the freedom to choose from a wide range of new products and services. Two, the computer hardware industry improves its products so quickly that a device that may be the top-of-the-line today may be mediocre one year later. For example,

---

6 Raymond, ix.
in the processor industry, Moore’s Law says that processors double in their number of transistors per integrated circuit every two years.7 This law continues to hold today as companies such as Intel and Advanced Micro Devices (AMD) relentlessly engineer new processor technology.

With the software industry, however, “change is measured in decades. The office suite, the 1980s killer application, wasn’t challenged until the 1990s with the introduction of the web browser and server.”8 In later sections, we will examine more carefully the open source model as a business plan and how some companies have embraced it to their advantage.

---

8 Raymond, ix.
2 History of open source

In this section, we provide a brief history of open source’s origins. The reader may be surprised to discover that the open source model is not a new phenomenon but one that has existed since the 1960s.

2.1 Pre-Unix: Multics

During the early 1960s, an OS called Multics (Multiplexed Information and Computing Service) was being developed by researchers at MIT, Bell Telephone Laboratories (BTL), and General Electric. It followed the well-received Compatible Time-Sharing System (CTSS) developed by Fernando J. Corbató at MIT. Both projects were attempts to implement a new concept called time-sharing. Mainframe computers during the 1960s were expensive and among the corporations, government laboratories, and universities that were able to afford them, each only owned a small number. Users had to sign-up to use their organization’s computer and long waiting periods were typical. However, when a user received the opportunity to use the computer, the entire system was at his disposal. He could use the machine as much as he wanted until his time slot expired, at which point another user’s time slot for the computer would begin.

This protocol of serving one user at a time on a first-come-first-serve was inefficient. Time-sharing was the solution that would allow numerous users to work “simultaneously” on a computer without having to wait for a long period. Users would connect via modem to a mainframe that would schedule short time slots for each user. Although this mechanism seemed to be like the original first-come-first-serve protocol, it differed in that rather than moving on to another user only upon full completion of his work, time-sharing completed partial amounts of work for many users in a given period of time; that is, a user used the computer’s resources for a

---

short time before they would be given to another user. Over time, the partial amounts of work would accumulate and many users would have their work completed. The key benefit was that users were unable to discern the time periods during which their work was put on hold so that other users’ work could progress.

CTSS achieved time-sharing and Multics was designed to extend its capabilities. Due to its complexity, however, Multics resulted in failure. Its developers wanted it to serve thousands of users simultaneously, but at its peak performance, Multics could barely serve three.\textsuperscript{10} Consequently, the “big-system” ideals that its designers had in mind were never realized. Frustrated with the lack of useful results, BTL withdrew its involvement from the project in March 1969.

\section*{2.2 Unix}

Ken Thompson was a BTL researcher who worked on the Multics project. During the summer of 1969, Thompson wrote a stripped-down version of Multics and called it Unics, which was eventually renamed Unix. Unix was developed with simplicity as a goal. Thomson and his colleagues avoided the complexity of Multics in order to create an OS whose leanness would lead to clean and understandable code. Since Unix was a novel idea, Thomson was apprehensive about the OS community’s response to it. He described his talk at the 1973 Association of Computing Machinery (ACM) Symposium on Operating Systems Principles in the following manner:

\begin{quote}
The audience was several hundred. I was pretty nervous. The response was the normal, polite applause. I don’t remember questions. As I recall I had seen the TENEX talk earlier [TENEX was an OS popular with users of the DEC-10] and
\end{quote}

adjusted my talk to show up some contrasts. I spent a lot of time on the shell and its implementation—trying to demonstrate how it was “just a program.” There was a question about the shell that completely missed the point. Perhaps I didn’t do a good job.\textsuperscript{11}

Despite his feelings, Thomson did very well. Peter Salus, author of \textit{A Quarter Century of UNIX}, writes that “[w]ithin six months of its delivery, the number of [Unix] installations had trebled.”\textsuperscript{12} After listening to Thomson’s talk, many researchers wanted to use Unix on their institution’s computers. From that day onward, Unix has grown to become one of the world’s most powerful OSs and is currently used in countless computing settings. In 1983, Thomson and Dennis Ritchie received the Turing Award (the Nobel Prize of computer science) for their work on Unix.

It is important to note that because Thomson created Unix as a BTL employee, its source code belonged to AT&T, BTL’s parent company. Like all corporations, AT&T was interested in profits and market dominance, so much that its activities caught the attention of the Department of Justice in the 1950s. In 1956, AT&T agreed to a consent decree that enjoined it and Western Electric (another subsidiary company belonging to AT&T) “‘from engaging in any business other than the furnishing of common carrier communications services….’”\textsuperscript{13} An important question that AT&T had to address was whether software should be considered a part of common carrier communications services. In 1956, AT&T’s attorneys lacked the hindsight to know that Unix would become very popular and they answered this question without much concern for the OS. Given AT&T’s strained relationship with the Justice Department, the

\textsuperscript{11} Peter Salus, \textit{A Quarter Century of UNIX} (Reading: Addison Wesley, 1994), 54. The quotation is from an interview with Thomson. Editorial comment in the quotation belongs to Salus.
\textsuperscript{12} Salus, 55.
\textsuperscript{13} Salus, 56-57.
attorneys opted for a conservative interpretation of the enjoinder that limited the company’s business to only telephones and telegraphs. Software as a business product was still novel and the attorneys did not want to risk upsetting the Justice Department by selling software. This decision was pivotal in allowing Unix to be released as open source software. Although every request for Thomson’s software had to be approved by AT&T’s legal department, the company was willing to provide the source code with few conditions and this willingness to release source code was necessary to begin the use of the open source model.

AT&T was further enjoined to rent patents and licenses to competitors without royalties. Again, it was the conservative interpretation by AT&T’s attorneys that pushed Unix into the realm of open source. Because AT&T was more concerned at the time with appeasing the Justice Department than making a profit from a piece of software that it did not fully appreciate, it was easy for the company to release Unix as open source software.

2.3 Berkeley Software Distribution (BSD)

University of California, Berkeley, was one of the many recipients of Thomson’s Unix. Robert Fabry, a computer science professor, wanted to bring Unix to Berkeley after hearing Thomson’s talk and with funds from Berkeley’s statistics, mathematics, and computer science departments, he purchased a computer to install and run the OS. Faculty members and students began using Unix and soon it became their favorite OS. Graduate students such as Bill Joy (co-founder of Sun Microsystems) and Chuck Haley modified parts of Unix and even sent bug fixes to AT&T. Their version of Unix was assembled with various tools and utilities as the Berkeley Software Distribution (BSD), and developers interested in obtaining BSD requested the software and its source code from Joy.
Over the last twenty years, BSD has evolved into specialized variants: FreeBSD, OpenBSD, and NetBSD. Each of these three variants emphasizes a particular technological aspect such as ease of installation, security, and hardware portability (that is, the ability to install the OS on numerous manufacturers’ hardware). The proliferation of BSD variants is an example of code forking: given a set of open source code, developers modify the code to create different versions of the original software. One way to understand code forking is to consider the original code as the “parent” code and the subsequent variants as “children” code.

As Steven Weber, author of The Success of Open Source, writes, “BSD and its offspring were not the only important Unix derivatives.” Companies such as IBM, Hewlett-Packard, and Sun Microsystems began to develop their own commercial versions in the early 1980s. In 1983, AT&T also began releasing its own proprietary version, System V, after recognizing the business potential of the software industry. AT&T was permitted to sell software because the 1956 consent decree was superseded by the resolution of a similar yet separate 1974 antitrust suit. The primary consequence of this antitrust suit was the dissolution of BTL into smaller companies, which meant that AT&T no longer had to release its OS at cost.

2.4 Linux

Just as a programmer should write his first piece of software as a “Hello, world” program, a historical account of Linux’s beginnings should quote an email from Linus Torvalds:

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

14 Weber, 43.
Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april [sic], and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

Like other programmers, Torvalds had found limitations with Minix (a small OS created by Andrew Tanenbaum for teaching purposes) and was motivated to create Linux to improve on Minix’s deficiencies. However, unlike Tanenbaum, he intended his project, Linux, to be “just a hobby”; during the early developmental stages of the OS, Torvalds wrote all of the code by himself. After writing the kernel (the core of an OS), Torvalds presented his code to the programming community and asked for their suggestions on improvements.

Programmers who conversed on the Minix newsgroup saw that Linux had the potential to become a powerful OS. Since Tanenbaum did not plan to improve Minix and the FSF’s OS, HURD, was not making headway, individuals who wanted to develop a new OS began writing code for Linux. Many programmers volunteered to work with Torvalds, and soon Linux became “big and professional.” New code updates were released almost every day and participants “contributed bug fixes, code improvements, and new features to Torvalds’s project.” Linux began to embody the principle of sharing code for the enrichment of an open source community, and the community’s cooperative work culminated in the release of Linux 1.0 in March 1994, three years after Torvalds announced the project.

Linux became so powerful that major corporations like Microsoft were becoming alarmed at its success. In internal company memos that are now called the “Halloween

---

17 Weber, 100.
18 Weber, 55.
19 Weber, 105.
Documents,” Microsoft’s Vinod Valloppillil admitted to management that “OSS [open source software] poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.”

Reference was being made to Linux. Although this memo was written in August 1998, Microsoft continues to struggle in its dealings with Linux. Microsoft has attempted to lure users away from Linux by disparaging it, but the company has recently acknowledged (only for internal purposes, but the information was leaked) that its efforts are at best ineffective and even “backfiring”; in the “Attitudes Towards Shared Source and Open Source Research Study,” “[r]atings for messages that were meant to be negative actually had a positive response among the respondents. […] [W]hen read what was supposed to be a negative OSS message about OSS and proprietary software having a similar TCO [total cost of ownership], nearly half (49%) of all respondents said that having heard this message they were now more favorable towards OSS.”

Rather than trying to discourage the use of Linux by calling it a “cancer” or “Pac-man-like,” Microsoft should aim to discourage the use of Linux by comparing it to its own OS, Windows, from a technical standpoint.

3 Public Licenses

Public licenses are at the heart of open source. For an individual to exercise the rights available from a piece of open source software, he must first agree to the terms and conditions of usage that allow him to copy, modify, and redistribute the software. The public license is the legal document that creates a contractual relationship between the open source software owner and the user. In a contractual setting, the owner is called the “licensor” because he grants a license for the software and its source code. Since the user receives the license from the owner, he is called the “licensee.”

Because the open source model is founded on the rights to copy, modify, and redistribute software, granting these rights is imperative for the model to function. The licensee must know that he can make as many copies of an open source application as he wants and that he is allowed to make changes to the software. In exchange for these freedoms, the licensee must agree to the specific conditions outlined in the license accompanying the software. For example, among the rules in the GNU General Public License (GPL), one of the most important is the following: if the licensee ever decides to release a modified version of an open source software (called a “derivative”) to the public, he must make the source code of the modified version available to his licensees. However, if he provides his software with the intention not to disclose its source code, the licensee effectively breaches the contract between him and the original licensor, since one of the conditions for using the open source software is making available the derivative’s source code. To close up software licensed under the GPL constitutes a breach of the license.

In the following sections, we will discuss from a legal perspective two public licenses that differ significantly from each other: the GPL and the BSD license. We will also contrast the

---

23 The word “public” is present to emphasize the availability of the software, its source code, and the freedoms to the public. One can drop the word “public” and still correctly call a public license simply a “license.” For our discussion, we will often call it a license.
GPL with a proprietary software license, the Microsoft Windows XP Professional End-User License Agreement (EULA).24

3.1 GNU General Public License (GPL)

When a licensor uses the GPL, he is “giving everyone the right to make unlimited copies of [his] work to freely redistribute it, and to make derivative works of it as long as those derivative works are also licensed under the GPL.”25 Created by Richard Stallman of the Free Software Foundation, the GPL is the most popular public license in the world. “According to Freshmeat, which calls itself ‘the Web’s largest index of Unix and cross-platform software,’ there are more than 19,000 GPL-covered software projects, and the GPL governs 68 percent of projects in the Freshmeat index.”26 Government officials, researchers, businesspersons, and hobbyists over the world use and modify software that is released under the GPL. Since the license is used extensively, Mark Radcliffe claims that “[t]he GPL has become the pivot point of a multibillion-dollar industry.”27 Two well-known programs, emacs (a text editor) and Linux, use the GPL as their license, and many other software applications use it to outline their conditions for usage, modification, and redistribution.

The GPL is a well-structured and clearly written legal contract. Its short length (exactly 2,970 words) and its readability allow a developer without legal training to understand it completely. Structurally, it begins with a preamble that states the motivating reasons to create

---

24 For convenience, we provide the GPL, the BSD license, and the Microsoft Windows XP Professional EULA in Appendix: Licenses.
26 Shankland, p. 2.
27 Shankland, p. 1.
open source software. It explains that whereas “[t]he licenses for most software are designed to take away your freedom to share and change it,” the GPL “is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.” This statement refers to proprietary software whose licenses often prohibit a user from copying and redistributing software. To ensure that licensors and licensees understand the meaning of the word “free” in the context of free software, the Preamble explains that it refers to freedom, not to an offer of gratis software. Both the licensor and licensee of a piece of open source software need to be aware that under the GPL, the licensor is allowed to distribute his open source software for a monetary price; the licensor may “charge for this service if [he] wish[es].” Licensors who distribute expensive open source software will want to recoup costs by charging for the software, and this provision allows for such practice. Licensees must realize that open source software does not have to be without a cost, although in practice most are gratis; otherwise, they might claim that open source software that is not gratis is illegitimately called “free.”

The preamble introduces the important concept of *copyleft*. Copyleft is the use of copyrights to ensure that derivatives of a GPLed software (called “work[s] based on the Program” in the GPL) remain free. The developer, on the basis of owning the copyrights to the software, allows users to copy, modify, and redistribute the software. The copyleft concept is applied strictly in the GPL: all derivatives must inherit the license conditions of their original software. By using copyleft, the repertoire of open source software increases as the possibility of closing up derivatives is precluded.

If copyleft is a de facto use of copyrights, then who owns the copyrights of the derivatives? Does the original licensor own them? Under U.S. copyright law, “the owner of
copyright [...] has the exclusive rights [...] to prepare derivative works based upon the copyrighted work[]” and can give others the permission to do likewise with his derivatives.\textsuperscript{28} In most circumstances, the owner of the copyright of the original work owns the copyrights to its derivatives. However, under U.S. copyright law, the right to derivatives “may be waived if the author expressly agrees to such waiver in a written instrument signed by the author.”\textsuperscript{29} Does the GPL act as a waiver instrument? Although it does not explicitly say that it is, a licensee can interpret section 2 of the GPL’s “Terms and Conditions for Copying, Distribution and Modification” as saying that it is a waiver instrument. It states that “it is not the intent of this section to claim rights or contest [the licensee’s] rights to work written entirely by [the licensee]; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.” This statement strongly suggests that licensor is not seeking to claim the copyrights of derivatives, and one can reasonably conclude that the licensor, through the GPL, expressly waives the rights to derivatives.

However, the issue of determining who owns the derivative copyrights is not as important as it may appear. The important fact is that the derivative is free: as long as it is free, anyone can exercise the rights to copy, modify, and redistribute the derivative. Under proprietary licenses, only the author can exercise these rights, but under the GPL, all licensees are allowed to exercise them. Thus, to own the copyrights of an open source software or its derivatives does not garner anyone any new rights that he would not have had if he did not own the copyrights. Copyrights are used simply as the means to ensure that all licensees can exercise the rights typically reserved for the author; they do not give any additional benefits in the context of open source software.

\textsuperscript{28} 17 U.S.C. §106(2).
\textsuperscript{29} 17 U.S.C. §106A(e)(1).
Since copyright ownership is only meaningful when rights are reserved, to seek ownership of the copyrights to the derivatives is unnecessary under the GPL.

Next, we turn to the GPL’s “viral clause,” a clause that prohibits the incorporation of GPLed software with non-GPLed software. In section 2 of the “Terms and Conditions,” it states that “when [the licensee] distribute[s] the same sections [i.e., parts of derivatives] 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.” This clause is interpreted to mean the following: a licensee who chooses to interweave a program that is released under the GPL with another program that is not released under the GPL must release the combined product under the GPL. The GPLed program “infects” the other non-GPLed programs, a phenomenon aptly described by the virus metaphor. Lest a licensee does not fully comprehend this prohibition, the last paragraph of the GPL provides intratextual reinforcement by stating that the license “does not permit incorporating [the licensee’s derivative] into proprietary programs.” Since the license of a proprietary program cannot be reconciled with the GPL, a licensee cannot interweave a GPLed software with a closed one. This last paragraph gives a particular instance of applying the viral clause.

The FSF justifies this clause using the premise that the combination of a GPLed program with a non-GPLed program is a derivative of the GPLed program. Since derivatives of GPLed software are free, the combination program must also be free. “A system incorporating a GPL-covered program is an extended version of that program. The GPL says that any extended
version of the program must be released under the GPL if it is released at all.” The FSF states the reasons for adhering to this premise: “[T]o make sure that users who get the software get the freedom they should have, and to encourage people to give back improvements that they make.” Neither of these reasons is sufficient to justify the consideration of the combination program as a derivative. The first reason is insufficient because users can still exercise their GPL rights on the GPLed part of the software even if the combination program is not considered a derivative (and thus not necessarily free under the GPL); the license conditions for the combination program would not affect the conditions of the GPLed program. As for the second reason, the viral clause is unlikely to “encourage” developers to enlarge the collection of open source software as the FSF purports. Rather, it can potentially frustrate and discourage developers. Sun Microsystems President Jonathan Schwartz explains that “[i]f people want to use the GPL and integrate [other software] with it, they have to adopt the proprietary license called the GPL[]. Basically it forces your hand. You don’t have any choice anymore.”

Although Schwartz’s claim that the GPL is proprietary is extreme, the clause does have a restrictive nature that ultimately prevents developers from solving problems that may require the incorporation of GPLed software with those that are not GPLed.

The FSF does not intend to prevent GPLed software from ever interacting with non-GPLed software; it permits a GPLed program and a non-GPLed program to be distributed together as long as they “communicate at arm’[’]s length.” However, the viral clause can be a potential source of litigation, since defining communication at arm’s length is difficult. Consider

31 Ibid.
33 “Frequently Asked Questions about the GNU GPL.”
the following discussion on differentiating between the permitted “communication at arm’s length” and the prohibited “incorporation”:

The difference between this and “incorporating” the GPL-covered software is partly a matter of substance and partly form. The substantive part is this: if the two programs are combined so that they become effectively two parts of one program, then you can’t treat them as two separate programs. So[,] the GPL has to cover the whole thing.

If the two programs remain well separated, like the compiler and the kernel, or like an editor and a shell, then you can treat them as two separate programs—but you have to do it properly. The issue is simply one of form: how you describe what you are doing.34

How does a developer precisely discern when two programs effectively become one? What sort of test could a court provide to help developers avoid breaching the GPL, and how useful would this test be? It is an established fact that a compiler and a kernel are different programs. However, many new technologies and concepts are blurring the line that helped distinguish traditional programs from each other. As the Internet becomes a central component of modern computing, substantive and form differences may become difficult to define. Distributed software (software whose components are placed on numerous machines over a network) could pose significant problems concerning the definition of software.

Another important issue to consider is whether the conditions of the GPL can be legally enforced by a court. However, to examine this issue, let us first explain what a license and a contract are. A license is “a unilateral abrogation of rights. The licensor has, by law, the ability to enforce certain rights against the licensee, and the license functions as a promise not to

34 “Frequently Asked Questions about the GNU GPL.”
enforce those rights.” Although David Gulbransen claims that a license and a contract are different, a license has the qualities of a contract and is considered as one. A contract is a legally binding agreement between an offeror (a person or an entity who offers something) and an offeree (a person or an entity who receives an offer) in which an offer has been made by the offeror, the offer has been accepted by the offeree, and there is an exchange between the offeror and offeree (called the consideration of the contract). All three of these elements are required for a contract to be considered valid. Consider the following scenario: a man offers to sell his computer for $1,000 to his neighbor. If the neighbor, who is the offeree in this scenario, agrees to pay this sum of money for the computer, then the offeror’s offer has been accepted. Notice that a computer is being exchanged for $1,000, so consideration exists. Given that the three necessary elements of a contract are present, the man and his neighbor have entered into a contract.

The GPL is both a license and a contract. The GPL is a license because it describes a licensor’s abrogation of his exclusive rights to copy, modify, and redistribute his software. It is also a contract because the three necessary elements that define a contract are present in the license. An offer exists because the licensor invites anyone who is interested in the software to obtain it under the GPL. Consideration exists because the licensee of the GPL must fulfill numerous conditions in exchange for the software, its source code, and the rights to copy, modify, and redistribute. For example, the licensee

may copy and distribute verbatim copies of the Program’s source code as [he] receive[s] it, in any medium, provided that [he] conspicuously and appropriately

35 Wacha, id.
publish[es] on each copy an appropriate copyright notice and disclaimer of
warranty; keep[s] intact all the notices that refer to this License and to the absence
of any warranty; and give[s] any other recipients of the Program a copy of this
License along with the Program.37

Note the words “provided that,” which indicate that the licensee may exercise the rights to copy
and redistribute a given GPL software as long as certain logistical procedures accompany the
exercising of the rights. The fulfillment of these procedures, with the software and the rights,
constitute the consideration of the contract created via the GPL: for the licensor’s software,
source code, and rights, the licensee fulfills the license conditions. Acceptance of the offer
occurs “by modifying or distributing the Program (or any work based on the Program), [and by
performing these activities the licensee] indicate[s] [his] acceptance of [the GPL] to do so, and
all its terms and conditions for copying, distributing or modifying the Program or works based on
it.”38 This form of acceptance is valid: “[c]ontracts can be accepted by performance, though
there is sometimes implied a requirement of an express warning that an act is required for
acceptance. The GPL, in Section 5, expressly provides that an act by the licensee will constitute
acceptance of the GPL’s terms.”39

Recent court cases validate this analysis. The parties of the following case assumed the
GPL to be a contract (The SCO Group claims that it is not a contract, but since it is still involved
in litigation with IBM, we will have to wait to see how the court rules on this claim):

Even when the GPL is implicated, the claims typically assume that the GPL is a
legal agreement and focus instead on whether the terms of the GPL were violated.

In MontaVista v. Lineo, for example, MontaVista alleged that Lineo distributed

37 Emphasis added.
38 Section 5 of the GPL.
39 Wacha, id.
computer programs copyrighted by MontaVista, and that, in such distributions by Lineo, “all references to MontaVista, including MontaVista’s copyright notice and contact information, [had] been removed.” While the GPL was implicated in the complaint, *neither party alleged that the GPL was not a valid agreement.* Instead, the question was whether—in addition to allegedly violating federal copyright law and other federal laws—Lineo violated *the terms of the GPL, which were accepted by both parties as valid and enforceable.* Similarly, MySQL’s counter-claim against Progress Software and NuSphere focused primarily on trademark infringement issues. However, MySQL alleged that NuSphere violated the GPL by failing to release source code to a GPL product. As in MontaVista v. Lineo, *both parties presumed the enforceability of the GPL;* the question presented, instead, was whether the defendant adhered to the GPL’s legal terms. Recently, however, The SCO Group incorporated into a court filing a direct claim that the GPL was illegal.\(^4^0\)

Jason B. Wacha provides an extensive analysis on the enforceability of the GPL and we use his article to guide our own assessment.\(^4^1\) According to Wacha, opponents of the GPL’s contractual validity claim that the lack of privity of contract and the vagueness of terms render the GPL unenforceable. Privity of contract is the direct relationship between two parties who have come to an agreement (called the “meeting of the minds”). Opponents assert that this privity does not exist between a licensor and a licensee of the GPL and that without it, the GPL is an unenforceable contract. Wacha rebuts this claim by stating that “[a]fter the introduction of the UCC [Uniform Commercial Code] and the various state statutes related to the UCC, privity of

\(^{40}\) Wacha, id. Emphases have been added and Wacha’s footnote references have been removed from the quotation.  
\(^{41}\) Wacha, id.
contract considerations have all but disappeared. In addition, some state courts have directly addressed and then rejected defenses based on lack of privity. Furthermore, general principals of contract law do not require an actual, subjective meeting of the minds between contracting parties. Consequently, the first argument against the validity of the GPL as a contract is not tenable given the UCC and state statutes.

The second argument based on the GPL’s vagueness of terms, however, is more difficult to debunk. A contract requires that key terms be clear and specific in their meanings. In Raffles v. Wichelhaus, 159 Eng.Rep. 375 (1864), it was held that if two contracting parties attached different meanings to a key term in a contract, the key term is ambiguous and no contract arises. Vague words can lead to ambiguity, and opponents of the GPL claim that two vague key terms exist in the GPL. Consider the following sentence of section 0 of the “Terms and Conditions”:

The “Program” […] refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: 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.

The first alleged vague key term is “copyright law.” Does the GPL invoke U.S. copyright law or another country’s? If more than one country’s copyright law could be referenced by this sentence, does that mean that the GPL contains a vague key word? Given that the GPL is used internationally, the answers to these questions are relevant to ensuring that the GPL is enforceable. “If there are licensees and licensors in different countries, for example, it may not be clear which copyright law applies, and a licensee may not truly understand her rights and

42 Wacha, id.
44 Section 0 of the GPL.
Furthermore, “[i]f the licensee and licensor are in different jurisdictions (and perhaps the server containing the downloaded code is in yet another jurisdiction), it may not be clear which jurisdiction’s copyright [law] is intended to apply.”  The second alleged vague key term is “work based on the Program,” which is the term used by the GPL to refer to derivatives. What exactly constitutes a derivative work? The U.S. Copyright Act of 1976 defines derivatives in the following manner:

A “derivative work” is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a “derivative work”.

The GPL explains derivatives in a similar yet subtly different manner:

[…] a “work based on the Program” means either the Program or any derivative work under copyright law: 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.

Note the differences between the two definitions. “The Copyright Act’s definition, for example, does not focus on whether one work ‘contains’ another work, or a portion of another work. Also, the Copyright Act predates the popularity of software programs.”  Even if we assume that this sentence refers to American copyright law, the GPL’s definition of a derivative may be broader.

---

45 Wacha, id.
46 Wacha, id.
48 Section 0 of the GPL.
49 Wacha, id.
than the definition established by U.S. federal circuit courts.\textsuperscript{50} Could these two key words affect the GPL’s enforceability?

As a rebuttal, Wacha explains that “[a] court will normally invalidate a contract for vagueness only when the terms are so unclear as to prevent the awarding of remedies for breach.”\textsuperscript{51} Although Wacha is correct in basing this principle on Kleinschmidt Div. of SCM Corp. v. Futuronics Corp., 363 N.E.2d 701 (1977), he is not completely accurate in making this particular claim. Endnote 199 of his article refers to this case and it summarizes the principle as the following: “[e]ven though one or more terms are left open a contract for sale does not fail for indefiniteness if the parties have intended to make a contract and there is a reasonably certain basis for giving an appropriate remedy.”\textsuperscript{52} However, the actual principle found in the case reads: “[u]nder the Uniform Commercial Code, if the parties have intended to contract, and if an appropriate remedy may be fashioned, a contract for sale does not fail for indefiniteness if terms, even important terms, are left open.”\textsuperscript{53} The difference between the two statements is that the latter is qualified with a reference to the UCC. Since Wacha shows in his article that the “[t]he UCC expressly does not cover such no-cost transactions,”\textsuperscript{54} in situations where a GPLed software is released without cost to the licensee, the Kleinschmidt principle will not necessarily hold.

Although Wacha should not use Kleinschmidt as the basis of his rebuttal, he does nonetheless rightly adhere to the principle that the existence of vague words should not necessarily void a contract. The reason why he should continue to use this principle is because “[t]he challenge based on an improper attempt at a legal definition [can fail] on several [other]
fronts: intent of the license, ability of parties to contract, and survivability of other provisions.” 55

Although the GPL’s definition of a derivative may vary from the one established by U.S. federal
circuit courts, courts should not nullify the GPL for its overbroad and software-minded
definition.

We have considered some of the implications of the GPL. Let us now turn to another
public license.

3.2 BSD License

Compared to the GPL, the BSD license is very short. Whereas the GPL restricts the closing up
of derivatives, the BSD license allows it. It is a liberal license that gives the licensee carte
blanche to do as he wishes, as long as he does not violate the one enumerated prohibition: he
cannot use the name of the licensor in endorsing derivatives unless written permission is granted
to do so. Under the second bullet of the license, the licensee can even withhold the source code
of the original and derivative software from others.

This license has been the source of heated debates between open source and free software
proponents. 56 Free software supporters argue that the license must have a clause that prohibits
the closing up of derivatives; according to them, such a clause is necessary to ensure that the
repertoire of free software grows. Although most open source supporters would wince at the
BSD license’s brevity and dearth of conditions, those who support it can claim that it still
permits the spread of open source software. Developers who wish to release their software’s
source code with the rights to copy, modify, and redistribute it can use the BSD license to
accomplish their distribution goals, just as they can use the GPL to do so. BSD license
supporters could also argue that the BSD license can encourage commercial developers who are

55 Wacha, id.
56 See section 1, “Introduction.”
normally skeptical of the GPL to consider using the open source model to distribute their software. That is, the BSD license can potentially attract developers who find the GPL too restrictive and have them consider using an open source license that gives them the option of closing up derivatives. Providing the opportunity for a developer to release as much software in the open form as he wants is better than preventing him from releasing any due to the strict conditions outlined in the GPL.

The open source proponents provide stronger reasons than their free software counterparts in the BSD license argument. Both groups ultimately want to encourage the use of public licenses to expand the collection of open source software. However, the free software community is unwilling to cater to the needs of developers who may be simultaneously interested in releasing source code and concerned about long-terms consequences involving the development of derivatives. The BSD license is a compromise between the GPL and proprietary licenses such as Microsoft’s EULA.

3.3 Comparison with Microsoft’s EULA

Microsoft’s Window XP Professional End-User License Agreement (EULA) is a proprietary license. It is used to license the Professional version of its flagship product, Windows XP, the world’s most popular OS. The Redmond, Washington, company licenses its products as closed software and uses a proprietary license to restrict a licensee’s rights and to disclaim as many legal liabilities as it can under existing laws. In 2004, however, the company acted unusually by disclosing the source code for FlexWiki (“a collaborative Web authoring software”), Windows Installer XML, and the Windows Template Library under the Common Public License (CPL).57

These source code disclosures were part of its Shared Source Initiative, a program developed to allow universities and governments to debug Microsoft’s software. Although open source proponents are currently debating the true openness of code released under the Shared Source Initiative, most of the company’s products are still released under licenses similar to the restrictive EULA.

From a legal perspective, the GPL is more beneficial to a licensor and a licensee than the EULA is. For instance, the GPL grants to the licensee the rights to copy, modify, and redistribute software licensed under it, whereas the EULA restricts these rights. Additionally, under the EULA, Microsoft spends millions of dollars in enforcing the license and in developing security technology to prevent licensees from violating the conditions of the license; the GPL licensor, on the other hand, spends money enforcing the GPL only under unusual circumstances such as litigation.

Consider the following table\textsuperscript{58} that compares the contents of the EULA and the GPL (the percentages are determined by paragraph section line counts):

<table>
<thead>
<tr>
<th>EULA</th>
<th>GPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The percentage of the license which limits your rights</td>
<td>45%</td>
</tr>
<tr>
<td>The percentage of the license which extends your rights</td>
<td>15%</td>
</tr>
<tr>
<td>The percentage of the license which limits your remedies</td>
<td>40%</td>
</tr>
</tbody>
</table>

\textit{Table 1. Comparing the contents of the GPL and the EULA. Percentages determined by paragraph section line counts.}

It is not surprising to discover that the primary emphasis of the EULA is to restrict the licensee’s rights and to limit the number of legal remedies available to him in cases of damages due to software malfunction. Microsoft tries to prevent users from examining the source code of its closed software and from depriving the company of its revenue through the copying and

It is also not surprising to discover that the primary emphasis of the GPL is the extension of the licensee’s rights. Half of the GPL’s content explains the licensee’s rights to copy, modify, and redistribute a GPLed software, and among other things, its other half describes provisions that protect the licensor who chooses to share source code with licensees. Since each of the licenses has different foundational principles concerning the way programs should be developed and shared, the EULA and the GPL place importance on different legal aspects related to the release of software.

One of the numerous prohibitions outlined in the EULA is the prohibition of making copies of Microsoft’s software: “[the licensee] may install, use, access, display and run one copy of the Product on a single computer, such as a workstation, terminal or other device.”59 Thus, to use the software, a licensee must agree that he will not make any copies of the software. However, under U.S. copyright law, this condition does not prevent the licensee from making one copy of the software “for archival purposes.”60 This exception includes even proprietary software like Windows whose licenses prohibit licensees from copying the software. The licensee is also prohibited from “reverse engineer[ing], decompil[ing], or disassemb[ling] the Product, except and only to the extent that it is expressly permitted by applicable law notwithstanding this limitation.”61 This limitation is important to Microsoft because technological procedures such as reverse engineering and decompiling can reveal the intermediary products that were used to create the final software. Although reverse engineering a piece of software does not produce the original source code, the results of the procedure can shed much light on how the software works and with these results, a knowledgeable developer

59 Bullet 1, Condition 1, Microsoft Windows XP Professional End-User License Agreement (EULA). Emphasis added.
61 Condition 5, Microsoft EULA.
might be able to reproduce software that mimics parts of the original Windows program. Given that these procedures can modify the closed nature of Windows, the licensee must agree not to use them except under special circumstances governed by law. One of these circumstances is the need to ensure interoperability between Windows and another third-party application. If a developer wants to create an application that interacts with Windows in a way not described by the Windows application programming interface (a set of tools provided by Microsoft to third-party developers who want to write applications for Windows), he may be justified in reverse engineering, decompiling, or disassembling Windows solely for the purpose of resolving interoperability issues. Nonetheless, such a developer would be wise to seek permission from Microsoft and to receive legal advice before undertaking any of these tasks. In the instance that Microsoft prohibits any of these tasks and the developer proceeds to perform them anyway, the developer may be risking an expensive and time-consuming litigation process.

Although Congress has legislated that a licensee can copy any program, including proprietary ones, for archival purposes, two potential obstacles stand in his way. The first obstacle is Microsoft’s “digital rights management [DRM] technology.”62 Some third-party affiliates provide “secure content” to users via the Internet and use DRM technology “to protect the integrity of their content […] so that their intellectual property, including copyright, in such content is not misappropriated.”63 Users must respect Microsoft’s and the third-party affiliates’ copyrights by not creating illegal copies of their software; however, in its effort to prevent such copying, Microsoft could also potentially use DRM technology to prevent users from making archival copies of their programs. The licensee must agree that “[o]wners of such Secure Content (“Secure Content Owners”) may, from time to time, request Microsoft to provide

---

62 Last bullet, condition 7, Microsoft EULA.
63 Last bullet, condition 7, Microsoft EULA.
security related updates to the Microsoft DRM components of the Product ("Security Updates")
that may affect [the licensee’s] ability to copy, display and/or play Secure Content through
Microsoft software or third party applications that utilize Microsoft DRM. For example,
suppose that in an attempt to curb illegal copying of its software, a secure content provider
creates a small closed program that prevents a user from making any copies of its software.
Under the abovementioned condition, the provider asks Microsoft to install this program on all
Windows machines containing secure content, even though it knows that the program infringes
on a licensee’s right to make an archival copy. Since the program is closed, Microsoft might not
even be aware of this violation and fulfills its obligations by installing it on Windows machines.
After the installation, a user discovers that he cannot make an archival copy of the provider’s
software. Thus, Microsoft and its affiliates can install such programs and potentially infringe on
a licensee’s right to make an archival copy. Among all the restrictive conditions that the EULA
may impose on a licensee, this secure content condition is especially restrictive and potentially
invasive.

The second obstacle is the Digital Millennium Copyright Act of 1998, which extends the
reach of copyright to prohibit the authoring and dissemination of technology that circumvents
measures taken to protect copyrights. The act cuts through the licensee’s right to make archival
copies because the right is supported by the fair use doctrine, a doctrine which cannot be used to
justify any circumvention of copyright protection technology. Consequently, if Microsoft
chooses to incorporate copyright technology into its Windows software (and it probably does),
the issue arises whether a licensee can create an archival copy of the software at all. Can a

64 Last bullet, condition 7, Microsoft EULA.
licensee make an archival copy of a software application where some of its components have copyright protection mechanisms and some do not?

Note that by implication, the right to modify Microsoft’s software is also prohibited. The only items available to the licensee is “the Microsoft software product […] , which includes computer software and may include associated media, printed materials, ‘online’ or electronic documentation, and Internet-based services […] “. The enumeration of these particular items implies that other items not enumerated, such as source code, are purposely withheld from the licensee. Microsoft wants to ensure that no third-party developer can modify (and publish) software released under the EULA, so it withholds the Windows source code. To further ensure that the EULA sufficiently prohibits the licensee from exercising rights that Microsoft wants to retain for itself, the license states that “Microsoft reserves all rights not expressly granted to [the licensee] in this EULA.” Since the rights to copy, modify, and redistribute the Windows software are not expressly granted in the EULA, Microsoft reserves these rights to itself.

Given the EULA’s restrictive conditions, the GPL is more beneficial to a licensee because it provides him with more rights. Although a licensee would never need to reverse engineer or decompile a GPLed software (since source code is always available), he is nonetheless free to carry out these procedures because the GPL does not prohibit them. Additionally, the licensee is free to copy, modify, and redistribute a GPLed software, whereas he is prohibited from such activities under the EULA. The GPL is also potentially more beneficial to the licensor than the EULA. Whereas the enforcement of the EULA and the development of DRM technology create additional costs for Microsoft, a typical GPL licensor spends nothing to

---

66 This sentence is found prior to the enumeration of the conditions in the EULA.
67 Lawyers would recognize this enumeration as the application of the *expressio unius est exclusio alterius* principle (“The explicit mention of one (thing) is the exclusion of another,” *Merriam-Webster Dictionary of Law* 1996).
68 Last bullet, condition 1, Microsoft EULA.
use the GPL. A licensor who uses the GPL has little incentive to protect his software from copying, modification, and redistribution, since these are expressly permitted activities under the GPL. The only occasions where a GPL licensor would spend money on using the license might arise when the licensor wants to confirm with a lawyer that the GPL is suitable for his software and to protect himself from other developers who might bring copyright or trademark infringement claims against him.\footnote{William R Della Croce, Jr, claimed that he, not Linus Torvalds, was the first person to use the trademark “Linux.” The need to have legal advice in such a situation is obvious. See \url{http://www.linux10.org/history/} for more details about this trademark infringement case and its resolution.}

The GPL is more likely to encourage greater software development than the EULA. With open source software, “[w]hen programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, [and] people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.”\footnote{“Welcome,” \textit{Open Source Initiative}, accessed 08 March 2005, \url{http://www.opensource.org/}.} The GPL contains conditions that facilitate the process of modifying existing code and redistributing new versions. The EULA, on the other hand, restricts the licensee from modifying the Windows software and precludes him from developing new programs based on it. Only Microsoft is able to create software from the Windows code, and the rate of software development does not increase by keeping the source code closed.

### 3.4 Conclusion

In this section, we analyzed the conditions of the GPL, the BSD license, and the EULA. The GPL grants to licensees the three rights that the EULA restricts: the rights to copy, modify, and redistribute software. In addition to granting these rights, it also ensures that the progeny of GPLed software remain free via the copyleft mechanism. Although the GPL should be
considered a valid and enforceable contract, a court could strike it down for containing vague key terms. The BSD license also grants to licensees the three rights that the GPL grants, but it permits licensees to close up derivatives; the GPL’s viral clause prevents licensees from closing up derivatives of GPLed software. Compared to the EULA, the GPL potentially gives greater benefits to the licensor and the licensee and as a result, licensors should consider using the GPL rather than a proprietary license.
4 Business Aspects

Given the popular misconception that open source software is available at no cost, it would not be surprising if some business people found the term “open source software company” to be an oxymoron. However, as we have explained in the first section of this paper, open source software need only be free in the sense of granting particular freedoms. A piece of open source software can be sold for a fee and still be considered open source. It is often the case that open source software is free of charge or available for a small sum of money; however, it does not have to be gratis nor inexpensive.

The open source model of software distribution is a viable business model, and there are many companies that make a profit by licensing open source software. Open source software has become such a profitable enterprise that IBM, Sun Microsystems, and Computer Associates decided to disclose source code of once-proprietary software along with freedoms similar to those granted by the GPL. Even Microsoft is trying to get a piece of the pie by releasing source code through its Shared Source Initiative.71 No commercial software company, especially those that license closed software, can ignore the market effect of open source software. Open source software is often cheaper than proprietary software, and this fact turns the heads of CTOs and CIOs. In fact, Forrester Research found that among 140 interviewed companies, the most popular reason for implementing Linux within their organizations was low acquisition cost.72 Open source software also reduces the risk of being locked into a particular supplier’s hardware or application; it can “place the customer in a much stronger negotiating position with its

71 Open source proponents wonder whether the Shared Source Initiative truly is open source.
commercial software providers.”73 Open source software companies threaten the dominant position that proprietary software companies have in the IT marketplace and are challenging traditional business models.

Some organizations, most notably Microsoft, argue that the open source model is dangerous for consumers and the businesses that sell software based on it. They claim that for consumers, security becomes an issue because the readily available source code allows malicious hackers to examine and exploit software vulnerabilities more easily than with a closed program. For businesses that license open source software, disclosing source code is destructive in the long-run. Microsoft Senior Vice President Craig Mundie claims that “[a] common trait of many of the [dotcom] companies that failed is that they gave away for free or at a loss the very thing they produced that was of greatest value—in the hope that somehow they’d make money selling something else.”74 Although these dotcom companies did not necessarily license open source software, the debate continues concerning the use of open source software in business settings, both as the technological framework to conduct one’s own business and as a product to license. Mundie and his followers must show why some companies thrive (and not merely survive) using open source as the central distribution model. However, the burden of proof is also on the open source companies to demonstrate to potential customers and stakeholders that their business philosophy is not inherently risky.

Textbooks and extensive commentaries have been written in pursuit of comparing open source and closed software business models. For our purposes, we limit our analysis in this section to two companies that develop and sell open source software: Red Hat and MySQL AB. We discuss some of their business strategies and conclude that the freedoms of the GPL need to

be at the forefront of their attempts to popularize their products. We also suggest that MySQL AB provides the ideal business model through its dual licensing system and that other open source software companies should seriously consider adopting the same licensing system to sell their products.

4.1 Case Study 1: Red Hat

Red Hat claims that it is “the world’s leading open source and Linux provider.” Based in Raleigh, North Carolina, Red Hat is well known for its Linux distribution, Red Hat Linux, which is used by many businesses, governments, and academic institutions. Its business customers include Amazon.com, AOL, Merrill Lynch, Credit Suisse First Boston, DreamWorks, Charles Schwab, Lehman Brothers, and Goodyear. Publicly owned, Red Hat has partnerships with major IT heavyweights such as IBM, Dell, HP, Oracle, Sun Microsystems, and Intel.

Red Hat’s flagship product is Red Hat Enterprise Linux (RHEL), an OS developed primarily for the commercial sector and that is licensed under the GPL. Red Hat offers variations of RHEL for different customer needs. Organizations use servers as their technology backbone and Red Hat markets two variants to cater to small and large entities alike. RHEL AS is recommended for large corporations that require powerful servers; for medium-sized companies, RHEL ES satisfies smaller workloads involving “file, print, mail, and web servers.” For desktop computers, Red Hat offers RHEL WS and Red Hat Desktop for users who want to run a Linux OS on their computer. The similarity among these products shows that Red Hat seeks to increase its market share by selling specialized versions of one core product; the

76 “Corporate Fact Sheet for Red Hat,” http://www.redhat.com/about/presscenter/presskit/fact_sheet.html. Other business customers and partners are listed at this website.
variations are similar to each other because they are the products of small-scale code forking.\textsuperscript{79} Using this variation strategy, Red Hat is able to improve all of its products when it improves the core set of code that underlies each version of RHEL. From a business perspective, internal code forking within Red Hat gives opportunities to meet customer needs that may be similar yet are different enough to require separate products.

In its report on the costs of using open source software, Forrester Research found that “[m]ore than half of the 140 companies surveyed rated the lack of support as a primary concern with open source.”\textsuperscript{80} Support has always been a major source of Red Hat’s revenue, but it will become even more important as Red Hat tries to address the issue of insufficient support. Since customers often require assistance to install and maintain a Red Hat product on their servers and workstations, Red Hat sells support subscriptions to meet these needs. Companies that have IT personnel who understand how to run Red Hat often purchase solely the software, but some organizations require support in using Red Hat; for these consumers, Red Hat offers packaged software, which is simply RHEL plus technical support. Red Hat also provides seminars to train IT professionals and certification programs for third-party technicians who want to leverage their qualifications.

Despite the success that Red Hat has encountered since its initial public offering in August 1999, it must emphasize the GPL’s freedoms to continue to maintain its market position and to differentiate itself from the proprietary software companies such as Microsoft. Red Hat shares similar marketing strategies with closed software companies and is trying to win over Windows and Unix users based on the benchmark features of RHEL. Although Red Hat’s strategy in comparing RHEL’s performance numbers to those of Windows and Unix is useful

\textsuperscript{79} For an explanation of code forking, see supra subsection 2.3, “Berkeley Software Distribution (BSD).”
and sure to increase the size of its customer base, the company must convince potential
customers to use RHEL because they can copy, modify, and redistribute the software. Red Hat
is beginning to resemble proprietary software companies in terms of its emphasis on support
subscriptions; it is “interesting to note that the pricing models being offered from [open source
software] distributors for maintenance and support closely resemble the models used by
commercial software providers like Microsoft.”81 Currently, Linux has had the benefit of having
a lower acquisition cost than its closed OS counterparts. However, if enough pressure is put on
closed software companies to offer more competitive prices, or if similar closed OSs surpass
RHEL in terms of benchmark figures, Red Hat may lose some of its customer base to proprietary
software companies. When this happens, Red Hat will need to emphasize the GPL freedoms that
distinguish its products from those of its proprietary competitors and convince customers that
these freedoms are worth purchasing.

Another reason why Red Hat must redefine its marketing strategy is because of the
debate concerning Linux’s total cost of ownership (TCO). Although it is clear that Linux
software is cheaper than other proprietary OSs, whether its TCO is also cheaper is less obvious.
Julie Giera of Forrester Research writes that “[t]he cost of software isn’t just the cost of Linux or
Windows—there still may need to be investments in systems management and monitoring tools,
either direct investments to purchase new products, or investments to upgrade/deploy Linux
support in existing systems management suites.”82 Running a Linux mainframe may require
more support from its vendor than running a Windows one, and if a company discovers that the
long-term cost of maintaining a Linux machine is costlier than maintaining a Windows machine,
it will opt to install Windows rather than Linux. Furthermore, fourteen companies with whom

---

Forrester Research had conversations “had a hard time finding qualified Linux personnel in the marketplace to support their Linux projects. When these companies did find third-party help, they had less leverage negotiating hourly rates than with other Windows consulting resources.”

Since there are fewer Linux consultants than there are Windows consultants, companies tend to pay more for Linux third-party support personnel, who are in high demand. Microsoft saw this TCO issue as an opportunity to weaken consumer confidence in Linux and has made the Forrester Research white paper available for public download. However, it is still unclear if Linux has a greater TCO than Windows because Forrester Research’s sample space in the white paper is relatively small; no definite conclusion should be made at this point.

An important question to consider is, how will Red Hat maintain its market share if the difference between the costs of acquiring RHEL and acquiring Windows becomes negligible, and if the TCO of owning RHEL becomes greater than that of owning Windows? Red Hat managers must realize that this hypothetical situation is possible. The key to ensuring that Red Hat maintains its favorable position in the marketplace is to highlight the benefits of the GPL. The strategy is essentially specialization: whereas Microsoft has maintained its position through closed software, Red Hat can differentiate itself from its competitor by providing the licensee with the rights to copy, modify, and redistribute RHEL. Red Hat cannot expect to fulfill its mission of becoming “the defining technology company of the 21st century” by marketing products that are similar to those of its competitors; it must offer its customers benefits that few of its competitors can or are willing to provide. Given that Microsoft is unlikely to adopt generous licenses like the GPL, Red Hat can maintain and increase its business by focusing on the GPL freedoms. It should continue to emphasize the performance benchmarks of RHEL.

---

compared to Windows, but it should also emphasize the freedoms as the primary reason to purchase RHEL.

If Red Hat decides to pursue such a marketing technique, it must convince companies that the process of modifying code is beneficial to the success of their business. Forrester Research advises companies not to change open source code:

Companies have understood for a long time that it costs much more money to support a custom or customized environment than it does to support a packaged software environment. Open source is no different. If an organization wants to limit its financial exposure, the best alternative is to avoid changing the code.

Remember, if you change it, you own it, and you support it. Just because a programmer can change the code, doesn’t mean they [sic] should.  

There is a lot of sense in telling IT personnel not to change code simply because they can. Changes to software come with a cost: managers must deliberate on the details of the change, programmers need to implement the changes in the code, and testers must stress-test and debug the new version of the software. However, for some companies, the right to modify software gives them opportunities to create specialized programs that suit their needs. Open source software generally gives companies more flexibility and control over their resources; they certainly do not have to change their open source software, but to have the opportunities to do so may minimize the dependency they have on suppliers, a benefit that they may want.

4.2   Case Study 2: MySQL AB

MySQL AB is a privately-owned Swedish open source database company. It is part of LAMP (Linux, Apache, MySQL, PHP/Perl/Python), which is “a fast growing open source enterprise

---

software stack [i.e., a set of software that meets major system needs]. MySQL, its main product, is the world’s most popular open source database software and has six million installations. According to a July 2004 SD Times study, MySQL was the third most popular database system among proprietary and open source software (see Figure 2 below). Its customers include the Associated Press, Dow Jones, Google, NYSE, Siemens, and the U.S. Census Bureau.

![MySQL is the #3 database among those polled](image)

**Figure 2.** July 2003 study by SD Times. MySQL was ranked third in highest number of deployments.

An important part of MySQL AB’s product development strategy is user stress-testing. Consider Figure 3 below. Likely called the “Virtuous Development Cycle” because MySQL AB generously provides the open source community with developmental versions of its software releases (source code included), this diagram shows that the interaction between users and MySQL AB is mutually beneficial. MySQL AB releases software to the programming

---

community for members to test the software in real-life settings; members provide comments on performance and bugs, and some even fix the problems for MySQL AB. MySQL AB then incorporates the comments and changes into its final version.

Figure 3. MySQL AB’s product development strategy involves user input.

By sharing software with users, MySQL AB has been able to maintain a higher level of code quality than proprietary database software companies. In December 2003, Reasoning, “the leading provider of automated software inspection (ASI) services[,] concluded that the code quality of MySQL ranks higher than commercial equivalents. Specifically, Reasoning found that MySQL code quality was [six times] better than that of comparable proprietary code, [that] MySQL benefits from the large communities of programmers who ‘battle test’ the code, and [that] MySQL benefits from users who not only report bugs, but track down their root cause and fix them.” ⁹⁰ Figure 4 below shows this information graphically.

---

MySQL AB uses a dual licensing system to allow both proprietary and open source developer groups to use MySQL. The dual licensing system gives a customer the option of choosing either the GPL or a commercial license for MySQL software. MySQL products released under the GPL are free-of-charge, but those that are released under the commercial license are sold for a price. Although MySQL software licensed under the GPL is gratis, some customers choose to pay the fee for the commercial license to avoid the copyleft and viral clauses of the GPL. That is, for derivatives of MySQL software licensed under the GPL, MySQL AB requires customers to release the source code if they release the derivative software to the public, regardless of whether the derivative is gratis or sold for a fee. Proprietary software developers have an incentive to pay for the commercial license because it will legally absolve any obligation for them to release the source code for derivatives based on MySQL software. Customers who choose the commercial license also gain the additional benefit of receiving commercial technical support, whereas those who choose the GPL must purchase support separately. Aside from the benefits of closing up derivatives and of receiving support, customers

---

of both licenses receive similar products since the software licensed under both the GPL and the commercial license have the same source code.

Open source companies should adopt and use MySQL AB’s dual licensing system. “[T]he dual licensing business model is becoming increasingly popular among open source companies as it paves the way for long-term financial viability. Other companies that offer dual licensing include Digium, OSAF, MandrakeSoft, Sleepycat Software, Technical Pursuit, Trolltech, and others.”

MySQL AB’s dual licensing system encourages open source development among individuals who are interested in sharing source code, but it also provides a realistic business solution to proprietary companies who want to use open source software in their own products. The GPL’s copyleft and viral clauses can deter proprietary companies from using software released under it. MySQL AB’s dual licensing system overcomes the GPL’s limitations and enables the company to cater to two market segments with one core product. Whereas Red Hat code forked its software, MySQL AB “license forked” and offers its products using the two different licenses.

4.3 Conclusion

The success experienced by companies such as Red Hat and MySQL AB demonstrates that open source products are worth selling. Such products are often bundled with technical support but are also available for purchase without support. As proprietary software companies such as Microsoft discover ways to counter open source success, Red Hat will need to place greater emphasis on the GPL freedoms associated with its Linux products to maintain its share of the OS market. This greater emphasis will also require Red Hat to convince its customers that despite

the costs and risks of modifying code, the right to do so will give greater flexibility to satisfy their ad hoc IT demands.

MySQL AB uses a dual licensing system that addresses the needs of proprietary and open source customers. The company gives the customer the option of choosing the GPL or a MySQL AB-specific commercial license for any of its products. Customers who want to withhold source code of MySQL derivatives should choose the commercial license, whereas others should opt to select the GPL. Whether motivated by financial considerations to reduce IT costs or by charitable goals to increase the repertoire of open source code, companies can use MySQL to run their own databases or to develop specialized products that will interact with MySQL software.
5 Conclusion and Further Research

As of April 20, 2005, SourceForge.net, “the world’s largest [o]pen [s]ource software
development website,” had 99,107 registered projects and 1,056,789 registered users. Such an
astoundingly high number of users indicates that the open source model of software distribution
is a popular and effective means of developing programs and applications. As more individuals
use and develop open source software, the legal and business issues of this cooperative
phenomenon will take on greater importance.

Given that this paper was limited in its scope, it did not address all of the issues related to
the open source model. Other questions of further study on open source are worth pursuing. A
popular debate among developers and governmental agencies concerns the usefulness of
software patents. How would open source software affect the issues involved in this debate?
What effects would the GPL have on the use of software patents?

Another area of further study is about Creative Commons. Creative Commons is a non-
profit organization that enables authors to offer their intellectual creations with “some rights
reserved.” Like the Free Software Foundation (who created the GPL), Creative Commons uses
U.S. copyright laws to establish a compromise between copyrights proper (“all rights reserved”) and the public domain (“no rights reserved”). It offers “layers” of copyrights to an author, who decides which layers will apply to his work. Although Creative Commons says that its licenses can theoretically be applied to software, it advises prospective licensors to use available public licenses such as the GPL instead. What issues would arise if a stubborn licensor used a

---

94 See http://creativecommons.org/.
95 “Learn More about Creative Commons,” Creative Commons, accessed 17 Mar 2005,
http://creativecommons.org/learnmore.
96 “Frequently Asked Questions,” Creative Commons, accessed 17 Mar 2005,
http://creativecommons.org/faq#faq_entry_3646. See the question, “Can I use a Creative Commons license for software?”
Creative Commons license anyway? Are Creative Commons licenses analogous to public licenses? Are they enforceable contracts?

We discussed in this paper the details of the GNU General Public License (GPL). We argued that it is very likely an enforceable contract and that it can potentially provide greater benefits to the licensor and licensee than a proprietary license such as the Microsoft End-User License Agreement (EULA). We also explored the issue of whether it is the original author or the derivative author who owns the copyrights to a derivative of a GPLed software. We concluded our legal discussion by comparing the GPL to the EULA and from the comparison found that the GPL seems to offer more benefits to the licensor and licensee.

From a business perspective, companies can use the GPL to license their software and still make a profit. As the successes of Red Hat and MySQL AB show, there is nothing inherently unprofitable about licensing software under the GPL. Many for-profit companies use the GPL or some other open source license to sell their products. In our analysis, we argued that it is important for open source companies to highlight the benefits of the GPL rights lest the companies become indistinguishable from proprietary software companies. An open source software company will need to differentiate its products from its proprietary software competitors by emphasizing the GPL rights to copy, modify, and redistribute software.
Bibliography

Books


Cases


Harald Welte v. Sitecom, District Court of Munich, 21 O 6123/04 (2004),


Electronic Books


Articles and Whitepapers


Zymaris, Con. “A Comparison of the GPL and the Microsoft EULA.”


Law Reviews

Online Newspapers


Websites

Creative Commons. http://creativecommons.org.
Appendix: Licenses

1 GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, 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.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect

97 “GNU General Public License,” Free Software Foundation, accessed 03 Feb 2005, http://www.fsf.org/licenses/gpl.html. To read FSF’s responses to frequently asked questions about the license, see http://www.fsf.org/licenses/gpl-faq.html.
making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: 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:

   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 scripts 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 public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year  name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it under certain conditions;
type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items—whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision'
(which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

2 BSD License\textsuperscript{98}

THE BSD LICENSE

The following is a BSD license template. To generate your own license, change the values of OWNER, ORGANIZATION and YEAR from their original values as given here, and substitute your own.

Note: The advertising clause in the license appearing on BSD Unix files was officially rescinded by the Director of the Office of Technology Licensing of the University of California on July 22 1999. He states that clause 3 is “hereby deleted in its entirety.”

Note the new BSD license is thus equivalent to the MIT License, except for the no-endorsement final clause.

\texttt{<OWNER> = Regents of the University of California}
\texttt{<ORGANIZATION> = University of California, Berkeley}
\texttt{<YEAR> = 1998}

In the original BSD license, both occurrences of the phrase “COPYRIGHT HOLDERS AND CONTRIBUTORS” in the disclaimer read “REGENTS AND CONTRIBUTORS”.

Here is the license template:

Copyright © <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.

3 Microsoft Windows XP Professional END-USER LICENSE AGREEMENT

MICROSOFT WINDOWS XP PROFESSIONAL END-USER LICENSE AGREEMENT

This End-User License Agreement (EULA) is for informational purposes only. There is no software accompanying the EULA.

IMPORTANT—READ CAREFULLY: This End-User License Agreement (“EULA”) is a legal agreement between you (either an individual or a single entity) and Microsoft Corporation for the Microsoft software product identified above, which includes computer software and may include associated media, printed materials, “online” or electronic documentation, and Internet-based services (“Product”). An amendment or addendum to this EULA may accompany the Product. YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA BY INSTALLING, COPYING, OR OTHERWISE USING THE PRODUCT. IF YOU DO NOT AGREE, DO NOT INSTALL OR USE THE PRODUCT; YOU MAY RETURN IT TO YOUR PLACE OF PURCHASE FOR A FULL REFUND.

1. GRANT OF LICENSE. Microsoft grants you the following rights provided that you comply with all terms and conditions of this EULA:

   • Installation and use. You may install, use, access, display and run one copy of the Product on a single computer, such as a workstation, terminal or other device (“Workstation Computer”). The Product may not be used by more than two (2) processors at any one time on any single Workstation Computer. You may permit a maximum of ten (10) computers or other electronic devices (each a “Device”) to connect to the Workstation Computer to utilize the services of the Product solely for File and Print services, Internet Information Services, and remote access (including connection

---

sharing and telephony services). The ten connection maximum includes any indirect connections made through “multiplexing” or other software or hardware which pools or aggregates connections. Except as otherwise permitted by the NetMeeting, Remote Assistance, and Remote Desktop features described below, you may not use the Product to permit any Device to use, access, display or run other executable software residing on the Workstation Computer, nor may you permit any Device to use, access, display, or run the Product or Product’s user interface, unless the Device has a separate license for the Product.

- Mandatory Activation. The license rights granted under this EULA are limited to the first thirty (30) days after you first install the Product unless you supply information required to activate your licensed copy in the manner described during the setup sequence of the Product. You can activate the Product through the use of the Internet or telephone; toll charges may apply. You may also need to reactivate the Product if you modify your computer hardware or alter the Product. There are technological measures in this Product that are designed to prevent unlicensed or illegal use of the Product. You agree that we may use those measures.

- Storage/Network Use. You may also store or install a copy of the Product on a storage device, such as a network server, used only to install or run the Product on your other Workstation Computers over an internal network; however, you must acquire and dedicate an additional license for each separate Workstation Computer on or from which the Product is installed, used, accessed, displayed or run. A license for the Product may not be shared or used concurrently on different Workstation Computers.

- Reservation of Rights. Microsoft reserves all rights not expressly granted to you in this EULA.

2. UPGRADES. To use a Product identified as an upgrade, you must first be licensed for the product identified by Microsoft as eligible for the upgrade. After upgrading, you may no longer use the product that formed the basis for your upgrade eligibility.

3. ADDITIONAL SOFTWARE/SERVICES. This EULA applies to updates, supplements, add-on components, or Internet-based services components, of the Product that Microsoft may provide to you or make available to you after the date you obtain your initial copy of the Product, unless we provide other terms along with the update, supplement, add-on component, or Internet-based services component. Microsoft reserves the right to discontinue any Internet-based services provided to you or made available to you through the use of the Product. This EULA does not grant you any rights to use the Windows Media Format Software Development Kit (“WMFSDK”) components contained in the Product to develop a software application that uses Windows Media technology. If you wish to use the WMFSDK to develop such an application, visit http://msdn.microsoft.com/workshop/imedia/windowsmedia/sdk/wmsdk.asp, accept a separate license for the WMFSDK, download the appropriate WMFSDK, and install it on your system.

4. TRANSFER—Internal. You may move the Product to a different Workstation Computer. After the transfer, you must completely remove the Product from the former Workstation Computer. Transfer to Third Party. The initial user of the Product may make a one-time transfer of the Product to another end user. The transfer has to include all component parts, media,
printed materials, this EULA, and if applicable, the Certificate of Authenticity. The transfer may not be an indirect transfer, such as a consignment. Prior to the transfer, the end user receiving the transferred Product must agree to all the EULA terms. No Rental. You may not rent, lease, lend or provide commercial hosting services to third parties with the Product.

5. LIMITATION ON REVERSE ENGINEERING, DECOMPIILATION, AND DISASSEMBLY. You may not reverse engineer, decompile, or disassemble the Product, except and only to the extent that it is expressly permitted by applicable law notwithstanding this limitation.

6. TERMINATION. Without prejudice to any other rights, Microsoft may cancel this EULA if you do not abide by the terms and conditions of this EULA, in which case you must destroy all copies of the Product and all of its component parts.

7. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.

- NetMeeting/Remote Assistance/Remote Desktop Features. The Product contains NetMeeting, Remote Assistance, and Remote Desktop technologies that enable the Product or other applications installed on the Workstation Computer to be used remotely between two or more computers, even if the Product or application is installed on only one Workstation Computer. You may use NetMeeting, Remote Assistance, and Remote Desktop with all Microsoft products; provided however, use of these technologies with certain Microsoft products may require an additional license. For Microsoft and non-Microsoft products, you should consult the license agreement accompanying the applicable product or contact the applicable licensor to determine whether use of NetMeeting, Remote Assistance, or Remote Desktop is permitted without an additional license.

- Consent to Use of Data. You agree that Microsoft and its affiliates may collect and use technical information gathered in any manner as part of the product support services provided to you, if any, related to the Product. Microsoft may use this information solely to improve our products or to provide customized services or technologies to you. Microsoft may disclose this information to others, but not in a form that personally identifies you.

- Internet Gaming/Update Features. If you choose to utilize the Internet gaming or update features within the Product, it is necessary to use certain computer system, hardware, and software information to implement the features. By using these features, you explicitly authorize Microsoft or its designated agent to access and utilize the necessary information for Internet gaming and/or updating purposes. Microsoft may use this information solely to improve our products or to provide customized services or technologies to you. Microsoft may disclose this information to others, but not in a form that personally identifies you.

- Internet-Based Services Components. The Product contains components that enable and facilitate the use of certain Internet-based services. You acknowledge and agree that Microsoft may automatically check the version of the Product and/or its components that you are utilizing and may provide upgrades or fixes to the Product that will be automatically downloaded to your Workstation Computer.
• Security Updates. Content providers are using the digital rights management technology ("Microsoft DRM") contained in this Product to protect the integrity of their content ("Secure Content") so that their intellectual property, including copyright, in such content is not misappropriated. Owners of such Secure Content ("Secure Content Owners") may, from time to time, request Microsoft to provide security related updates to the Microsoft DRM components of the Product ("Security Updates") that may affect your ability to copy, display and/or play Secure Content through Microsoft software or third party applications that utilize Microsoft DRM. You therefore agree that, if you elect to download a license from the Internet which enables your use of Secure Content, Microsoft may, in conjunction with such license, also download onto your computer such Security Updates that a Secure Content Owner has requested that Microsoft distribute. Microsoft will not retrieve any personally identifiable information, or any other information, from your computer by downloading such Security Updates.

8. NOT FOR RESALE SOFTWARE. Product identified as "Not for Resale" or "NFR," may not be resold, transferred or used for any purpose other than demonstration, test or evaluation.

9. ACADEMIC EDITION SOFTWARE. To use Product identified as "Academic Edition" or "AE," you must be a "Qualified Educational User." For qualification-related questions, please contact the Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399 or the Microsoft subsidiary serving your country.

10. EXPORT RESTRICTIONS.

You acknowledge that the Product is of U.S. origin and subject to U.S. export jurisdiction. You agree to comply with all applicable international and national laws that apply to the Product, including the U.S. Export Administration Regulations, as well as end-user, end-use, and destination restrictions issued by U.S. and other governments. For additional information see http://www.microsoft.com/exporting/.

11. LIMITED WARRANTY FOR PRODUCT ACQUIRED IN THE US AND CANADA.

Microsoft warrants that the Product will perform substantially in accordance with the accompanying materials for a period of ninety days from the date of receipt.

If an implied warranty or condition is created by your state/jurisdiction and federal or state/provincial law prohibits disclaimer of it, you also have an implied warranty or condition, BUT ONLY AS TO DEFECTS DISCOVERED DURING THE PERIOD OF THIS LIMITED WARRANTY (NINETY DAYS). AS TO ANY DEFECTS DISCOVERED AFTER THE NINETY (90) DAY PERIOD, THERE IS NO WARRANTY OR CONDITION OF ANY KIND. Some states/jurisdictions do not allow limitations on how long an implied warranty or condition lasts, so the above limitation may not apply to you.

Any supplements or updates to the Product, including without limitation, any (if any) service packs or hot fixes provided to you after the expiration of the ninety day Limited Warranty period are not covered by any warranty or condition, express, implied or statutory.
LIMITATION ON REMEDIES; NO CONSEQUENTIAL OR OTHER DAMAGES. Your exclusive remedy for any breach of this Limited Warranty is as set forth below. Except for any refund elected by Microsoft, YOU ARE NOT ENTITLED TO ANY DAMAGES, INCLUDING BUT NOT LIMITED TO CONSEQUENTIAL DAMAGES, if the Product does not meet Microsoft's Limited Warranty, and, to the maximum extent allowed by applicable law, even if any remedy fails of its essential purpose. The terms of Section 13 below (“Exclusion of Incidental, Consequential and Certain Other Damages”) are also incorporated into this Limited Warranty. Some states/jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. This Limited Warranty gives you specific legal rights. You may have others which vary from state/jurisdiction to state/jurisdiction. YOUR EXCLUSIVE REMEDY. Microsoft’s and its suppliers’ entire liability and your exclusive remedy shall be, at Microsoft’s option from time to time exercised subject to applicable law, (a) return of the price paid (if any) for the Product, or (b) repair or replacement of the Product, that does not meet this Limited Warranty and that is returned to Microsoft with a copy of your receipt. You will receive the remedy elected by Microsoft without charge, except that you are responsible for any expenses you may incur (e.g. cost of shipping the Product to Microsoft). This Limited Warranty is void if failure of the Product has resulted from accident, abuse, misapplication, abnormal use or a virus. Any replacement Product will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. Outside the United States or Canada, neither these remedies nor any product support services offered by Microsoft are available without proof of purchase from an authorized international source. To exercise your remedy, contact: Microsoft, Attn. Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399, or the Microsoft subsidiary serving your country.

LIMITED WARRANTY FOR PRODUCT ACQUIRED OUTSIDE THE US OR CANADA. FOR THE LIMITED WARRANTIES AND SPECIAL PROVISIONS PERTAINING TO YOUR PARTICULAR JURISDICTION, PLEASE REFER TO YOUR WARRANTY BOOKLET INCLUDED WITH THIS PACKAGE OR PROVIDED WITH THE SOFTWARE PRODUCT PRINTED MATERIALS.

12. DISCLAIMER OF WARRANTIES. The Limited Warranty that appears above is the only express warranty made to you and is provided in lieu of any other express warranties (if any) created by any documentation, packaging, or other communications. Except for the Limited Warranty and to the maximum extent permitted by applicable law, Microsoft and its suppliers provide the Product and support services (if any) AS IS AND WITH ALL FAULTS, and hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of reliability or availability, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the Product, and the provision of or failure to provide support or other services, information, software, and related content through the Product or otherwise arising out of the use of the Product. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE PRODUCT.
13. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, PUNITIVE, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE PRODUCT, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT OR OTHER SERVICES, INFORMATION, SOFTWARE, AND RELATED CONTENT THROUGH THE PRODUCT OR OTHERWISE ARISING OUT OF THE USE OF THE PRODUCT, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

14. LINKS TO THIRD PARTY SITES. You may link to third party sites through the use of the Product. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.

15. LIMITATION OF LIABILITY AND REMEDIES. Notwithstanding any damages that you might incur for any reason whatsoever (including, without limitation, all damages referenced above and all direct or general damages), the entire liability of Microsoft and any of its suppliers under any provision of this EULA and your exclusive remedy for all of the foregoing (except for any remedy of repair or replacement elected by Microsoft with respect to any breach of the Limited Warranty) shall be limited to the greater of the amount actually paid by you for the Product or U.S.$5.00. The foregoing limitations, exclusions and disclaimers (including Sections 11, 12 and 13 above) shall apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

16. U.S. GOVERNMENT LICENSE RIGHTS. All Product provided to the U.S. Government pursuant to solicitations issued on or after December 1, 1995 is provided with the commercial license rights and restrictions described elsewhere herein. All Product provided to the U.S. Government pursuant to solicitations issued prior to December 1, 1995 is provided with “Restricted Rights” as provided for in FAR, 48 CFR 52.227-14 (JUNE 1987) or DFAR, 48 CFR 252.227-7013 (OCT 1988), as applicable.
17. APPLICABLE LAW. If you acquired this Product in the United States, this EULA is
governed by the laws of the State of Washington. If you acquired this Product in Canada, unless
expressly prohibited by local law, this EULA is governed by the laws in force in the Province of
Ontario, Canada; and, in respect of any dispute which may arise hereunder, you consent to the
jurisdiction of the federal and provincial courts sitting in Toronto, Ontario. If this Product was
acquired outside the United States, then local law may apply.

18. ENTIRE AGREEMENT. This EULA (including any addendum or amendment to this
EULA which is included with the Product) are the entire agreement between you and Microsoft
relating to the Product and the support services (if any) and they supersede all prior or
contemporaneous oral or written communications, proposals and representations with respect to
the Product or any other subject matter covered by this EULA. To the extent the terms of any
Microsoft policies or programs for support services conflict with the terms of this EULA, the
terms of this EULA shall control.

19. The Product is protected by copyright and other intellectual property laws and treaties.
Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the
Product. The Product is licensed, not sold.

[Additional French text follows and has been omitted.]