Software Licenses: Taxonomy and Analysis

Daniel Loreto, Adam Oliner, and Ram Woo

May 10, 2004

Abstract

When software is distributed, it is generally accompanied by a license that enumerates the rights retained and released by the author or distributor. Use of the software is ostensibly predicated by acceptance of this license. Therefore, a better understanding of licenses is an asset both to authors and users of software. This paper presents a general taxonomy of both open-source and proprietary software licenses. We examine several major licenses in detail, and discuss the relative merits of the available options. Additionally, we have created an interactive, online License Selection Engine that assists in selection of a license based on the desired license properties.

Disclaimer: We are not lawyers, and this is not legal advice. We present this work as is, including any and all factual errors, and accept no responsibility or liability for whatever you might choose to do with it. The opinions expressed herein are solely those of the authors of this paper.

Contents

List of Figures **2**

1 Introduction **3**

2 Taxonomy **4**
2.1 Open-Source Licenses 4
2.2 Proprietary Licenses 6
2.2.1 Hardware Permitted 6
2.2.2 Network Use 7
2.2.3 Activation Required 7
2.2.4 Backups Permitted 7
2.2.5 Derivative Work 8
2.2.6 Produced Work 8
2.2.7 Resell Policy 8
2.2.8 Export Restrictions 8
2.2.9 Warranty 9
2.2.10 Liability 9
2.2.11 Jurisdiction 9
2.2.12 Miscellaneous 9

3 Discussion **9**
3.1 GPL 13
3.2 Microsoft EULA 13
3.3 Hybrid Model 14
3.3.1 Why Open? 14
3.3.2 Why Closed? 14
3.3.3 Compromise 15

4 License Selection Engine **15**
4.1 Technical Details 16
4.2 Related Work 16

5 Contributions **19**

6 Acknowledgments **19**

References **20**

List of Figures

1 Broad License Selection 17
2 Specific License Selection 18
3 Selected Licenses 19

1 Introduction

Selecting a license is a major decision for any software project. The type of license you choose may influence the popularity, security, reliability, quality, feature-set, and user-base of your software. In order to pick the correct license, you should first understand the kinds of rights that many licenses tend to address, and what licenses are frequently used. You should then consider the goals of your project, and the consequences of picking a particular license. The purpose of this document is to help you select a software license[1].

There are two important things to understand about software licenses before we continue. First of all, United States copyright law grants the creators of original works certain rights that are retained implicitly. Specifically, since 1976, copyright is bestowed upon "works of original authorship fixed in tangible media," including software. It seems that even the experts are not sure what happens when you try to forfeit these basic rights [14], but the consensus seems to be that it is not possible to explicitly and voluntarily place your software in the public domain. One important entity holding the opposite view is the Creative Commons[2] [2]. Most agree that, if you want your work in the public domain, you should just distribute your software with an extremely permissive license like the MIT License. You are using a license to grant other people the right to use your software under the conditions you specify.

The second important thing to understand is that a license grants rights to the recipient, it does not restrict the author of the software. For example, just because you distribute your source code under the GPL [5], this does not mean that you cannot include that code in proprietary software; it simply means that other people cannot. You can change your mind about the license under which you wish to distribute your software, but this can be messy. Read this document carefully and make your decision a permanent one.

The highest level of our classification hierarchy, and the first thing your should think about when picking a license, is whether the source should be open, closed, or some combination thereof. In other words, do you want to provide your customers with access to your source code under whatever restrictions are specified in the license? In this paper, we argue that open-source is typically a better choice, and that even proprietary vendors should expose the source for certain key parts of their software. We argue that those portions of the code responsible for compatibility and security are the most important ones to expose, and that this will be beneficial both to the vendor and to the consumers.

To help you select a license, we have created a license selection engine that can be accessed online and is based on our license classification table. The License Selection Engine [12], as we have called it, may serve as a guide to help narrow the field of license choices. After that, careful attention should be paid to specific details, and major projects should consult a legal authority.

This paper, therefore, presents a taxonomy of both open-source and closed-source software licenses, examines two major licenses in detail, suggests a strategy for opening selected portions of project code for the benefit of everyone involved.

[1]As we made clear in the disclaimer, we are not offering legal advice, nor are we qualified to do so. What we are presenting here is our own understanding of the universe of software licenses, and our personal opinions on the subject. While we are not lawyers, we are experienced and well-trained computer scientists. As such, the technical aspects of this report are more reliable than the legal aspects. Before making legal decisions, you should contact a lawyer.

[2]Their policy on how to put something in the public domain can be found at:
http://creativecommons.org/license/publicdomain-direct

2 Taxonomy

In this section, we provide a taxonomy of both open-source and proprietary software licenses. The taxonomy analyzes a number of different software licenses and provides a breakdown of their major features in a succinct, easy-to-read format. These features were chosen based on their importance to, and the needs of, software developers.

2.1 Open-Source Licenses

An open-source license facilitates the evolution of software by requiring software covered under the license to distribute all program files associated with the software. In this section, several open-source software licenses were analyzed including: Public Domain, Academic Free License, Common Public License, Lesser General Public License, Open Software License, Open Software License, General Public License, Modified BSD License, IBM Public License, Apple Public Source License, as well as the INTEL Open Source License [7]. This list is repeated in Table 2.1. These software licenses were chosen as a representative cross-section of the vast number of available open-source licenses. We feel that these licenses should be able to cover the majority of software developed by individual programmers and small software companies. There are a variety of features that can be measured for each type of license. These range from whether the license text needs to be included upon redistribution to the type other software that ones work can be included.

Each of the chosen licenses were analyzed for a number of features and the results are displayed in Tables 2.1, 2.1, and 2.1.

The following list explains the meanings of the various license features:

- *Copyright notice must be attached?* Does the license require users to include the copyright notice in all redistribution of the source code?
- *Same license for larger work?* Does the license require that all larger software packages containing all or part of the source code (including modifications) fall under the said license?
- *Users can modify the code?* Are users granted the freedom to modify all or parts of the source code?
- *Users can create derivative works?* Does the license grant the user the freedom to modify or use the source code as part of a larger distribution or another software package?
- *Users can ask money for redistribution of the software code, part or whole?*Does the license allow users to ask for money when redistributing any part of the source code?
- *GPL compatible?* Is the license compatible with the GNU General Public License?
- *User can create binary applications?* May the code be compiled into a binary application runnable by a computer?
- *Must distribute modified source code?* Does this license require users to distribute source code for any modified versions of the software?
- *Copyleft?* All users have the right and freedom to use, modify, and redistribute the program's source code or *any programs derived from it.* [3]

Public Domain (PD)
Academic Free License (AFL)
Common Public License (CPL)
Lesser General Public License (LGPL)
Open Software License (OSL)
General Public License (GPL)
Modified BSD License (mBSD)
IBM Public License (IBM PL)
Apple Public Source License (APSL)
INTEL Open Source License (INTEL OSL)

Table 1: A table summarizing the open-source licenses considered in this paper.

	PD	*AFL*	*CPL*	*LGPL*	*OSL*	*GPL*
Copyright Notice Must Be Attached?	N	Y	Y	Y	Y	Y
Same License For Larger Work?	N	N	N	N	Y	Y
User Can Modify the Code?	Y	Y	Y	Y	Y	Y
User Can Create Derivative Works?	Y	Y	Y	Y	Y	Y
User Can Ask Money For His/Her Derivative Works?	Y	Y	Y	Y	Y	N
User Can Ask Money For Redistribution of Software Code?	Y	Y	-	-	Y	N
GPL Compatible?	N	N	N	Y	N	Y
User Can Create Binary Applications?	Y	Y	Y	Y	Y	N
User Must Distribute Modified Source Code?	N	N	Y	Y	Y	Y
Copyleft?	N	N	-	Y	Y	Y

Table 2: Taxonomy Table: Part I: Summary of the differences and similarities between several open-source licenses. A '-' indicates that it was not clear from the license what the stance is on that issue.

	Mozilla	*Artistic*	*mBSD*	*Apache*
Copyright Notice Must Be Attached?	Y	Y	Y	Y
Same License For Larger Work?	N	N	N	N
User Can Modify the Code?	Y	Y	Y	Y
User Can Create Derivative Works?	Y	Y	Y	Y
User Can Ask Money For His/Her Derivative Works?	Y	Y	Y	Y
User Can Ask Money For Redistribution of Software Code?	-	N	Y	N
GPL Compatible?	N	Y	Y	N
User Can Create Binary Applications?	Y	Y	Y	Y
User Must Distribute Modified Source Code?	Y	Y	N	-
Copyleft?	Y	Y	N	N

Table 3: Taxonomy Table: Part II: A continuation of Table 2, summarizing the differences and similarities between several open-source licenses. A '-' indicates that it was not clear from the license what the stance is on that issue.

	IBM PL	*Apple PSL*	*INTEL OSL*
Copyright Notice Must Be Attached?	Y	Y	Y
Same License For Larger Work?	Y	N	N
User Can Modify the Code?	Y	Y	Y
User Can Create Derivative Works?	Y	Y	Y
User Can Ask Money For His/Her Derivative Works?	Y	-	-
User Can Ask Money For Redistribution of Software Code?	-	-	-
GPL Compatible?	N	N	Y
User Can Create Binary Applications?	Y	Y	Y
User Must Distribute Modified Source Code?	-	Y	N
Copyleft?	-	-	N

Table 4: Taxonomy Table: Part III: A continuation of Table 3, summarizing the differences and similarities between several open-source licenses. A '-' indicates that it was not clear from the license what the stance is on that issue.

2.2 Proprietary Licenses

A license is deemed proprietary if the source is closed, either because it is not provided in human-readable form, or because you're not allowed to use it to make derivative works. In the following section we analyze several proprietary software licenses. Namely, Microsoft Windows XP, Apple OS X, Microsoft Office, Adobe Photoshop, Macromedia Dreamweaver, Sun Java, and Winzip. These licences were selected because we felt they are representative of a wide variety of proprietary software that is in common use. Windows XP and OS X are two common operating systems; Office, Photoshop, and Dreamweaver are content-editing applications used in many business environments; Java and Winzip are two software utilities that are widely distributed.

For each of these applications, we analyze several features that we consider the most relevant. We explain each of them in detail below. Tables 2.2.12, 2.2.12, and 2.2.12 summarize the results of our analysis.

2.2.1 Hardware Permitted

Hardware permitted describes the type of hardware on which you are allowed to run the software. The most common enforcement is to allow the user to install the software on one main computer and on one laptop computer. The software is to be used by the user only and, if the user gets a new computer, he or she can transfer the software, provided it is removed from the old computer.

By far the most restrictive licenses in this case were the OS licenses. Apple specifies that the computer must be Apple-labelled. So that means you can only run the software on a computer bought from Apple. If you made a clone, you would not be allowed to run the software there. Microsoft's license limits the number of processors that the computer can have and specifies that the software is to be bundled with a specific computer. That is, if a user gets a new computer, he or she must buy a new license. The restriction has other implications that are not so clear. When does a computer cease to be the same computer and become a new one? If you install new memory, certainly the answer should be that the computer is the same, but what if you upgrade the processor, or switch the motherboard?

The licenses for the software utilities we analyzed were more lenient. Java's license [6] allows the software to be used on an unlimited number of computers. However, it is also given for free, so it is only logical that this would be the case. WinZip's [16] license allows the user to install it on an unlimited number of computers as long as only one person uses the software. Alternatively, the software can be used by an unlimited number of users as long as it is only installed on one computer.

Of all the options provided, we felt that WinZip's was the best for commercial proprietary software. It strikes the right balance between providing flexibility for the user and disallowing unlimited distribution. Both of the choices it provides are very logical: a consumer can buy the software for their own personal use and they can use it on whatever computer they want, or the consumer can buy it for the house computer so the whole family can use it.

2.2.2 Network Use

Besides restricting the hardware in which the software can be used, companies also restrict how the software can be used through a network. Most of them allow network use through an alternative license option, in which there is a copy of the software running on a server. The server must have a license; as well as every client that accesses the software on the server.

The one exception were the operating systems, which do not allow network use. That is, a client computer cannot run the operating system from a server, it must be locally installed. Because operating systems also provide network services, some companies also place restrictions on the use of those services. Windows, for example, only allows a maximum of 10 concurrent connections to use the network services provided by the OS. Apple's OS X [8], on the other hand, places no restrictions on the number of connections that may use the network services provided by the OS software.

2.2.3 Activation Required

Some software requires a registration or activation procedure before being fully functional. In most cases, this registration is optional. In the licenses we studied, there were two main exceptions: WinZip and Microsoft's products.

WinZip's requirement is understandable. It is only enforced when you have the free version of the software, which is intended to be used during a demonstration period. After the demonstration period the user can opt to buy the software and get an activation key that removes all restrictions.

Microsoft's case, is different. They require you to register and activate the software after you have bought the software and paid for the right to use it. The activation can ask any sorts of questions, including personally identifiable information. This is, in fact, contradicting other parts of the license which states that, although you agree Microsoft will collect information from you, it will do so in a manner that is not personally identifiable. We feel this mandatory activation is undesirable as it places an additional burden on the customer and raises privacy concerns. If the intended goal is to prevent privacy, there are other means that are effective.

2.2.4 Backups Permitted

Although companies generally try to prevent consumers from making copies of the software, they generally allow making copies for backup purposes. The most common option is to allow customers to make one backup copy as long as the copyright notices are included in the backup.

Microsoft Office was more restrictive, and did not allow any backup copies to be made. It considered the original media to be a backup copy, which in our opinion, is a ludicrous statement because it prevents users from making backups of updated state in their hard-drive. For example, if the user installs the software, makes changes to the configuration, and installs a new patch, he or she would not be able to make a backup of the resulting state.

Dreamweaver, Java and WinZip were more lenient. Dreamwever allowed for a "reasonable number" of backup copies to be made. Java and WinZip allowed for an unlimited number of copies. In our opinion, all companies should take this approach, and allow the customers to make as many backups as necessary to safe-guard their software and data. Especially since, as we explain in the warranty section, companies generally don't make any warranty with respect to the media the software is provided in, and if they do, it is generally limited to only 90 days.

2.2.5 Derivative Work

Unlike open-source software, basically all proprietary software prohibits any derivative work from being created. Specifically, companies generally prohibit, alterations, modifications and any sort of reverse engineering, decompilation and dissasembly.

Most licenses stated this terms in different but similar wording. One license, however, struck us as peculiar on the way it phrased the terms. Macromedia [9] stated that the user may not "reduce the Software to a human perceivable form". We believe the intended interpretation is the same as for the other software, but under a more lenient interpretation it could even restrict a user from writing about how a program works.

2.2.6 Produced Work

Produced work refers to the restrictions placed on content created to the use of the software. This section does not apply to the operating systems, but applies to the other software we studied.

Some of the software placed no restriction on the work you could create with it. However, some, like Microsoft Office [10] and Adobe Photoshop [1], placed restrictions on the content that could be created. Office did not allow for obscene or scandalous material to be created using clip-art elements. Photoshop prohibits the use of included stock for pornographic or illegal material.

In our opinion, no restrictions should be placed on the work users create using a piece of software. Users should be able to use the software they legally paid for to produce any kind of work they want. Whether that work raises legal issues or not, should not be the concern of the software company, especially since they claim no liability whatsoever.

2.2.7 Resell Policy

With most physical products, you are allowed to resell the product to another customer and transfer all the accompanying rights. Software companies, however, impose restrictions on how you can resell the software. Many of them, enforce a one-time permanent transfer policy. Which means that only the original licensee is allowed to resell it; once he does so, the customer that bought the license from him is not allowed to resell it – ever. Other companies allow a permanent transfer policy, which basically means that whoever acquires the resold license, can resell it again if he or she so desires. In both cases, the seller must get rid of all his copies of the software and the buyer must agree to all the terms of the license.

In our opinion, a permanent transfer policy is preferable that the one-time permanent transfer policy. Since it allows indefinite reselling and still guarantees that the license terms are enforced.

2.2.8 Export Restrictions

Because all the companies that we studied were US based, all of them licensed their software according to US export restrictions. Namely, they state that the software may not be exported or re-exported (a) into (or to a national or resident of) any U.S. embargoed countries (currently Cuba, Iran, Iraq, Libya, North Korea, Sudan, and Syria) or (b) to anyone on the U.S. Treasury Departments list of Specially Designated Nationals or the U.S. Department of Commerce Denied Persons List or Entity.

Software made outside the US is not subject to these restrictions.

2.2.9 Warranty

Software is probably one of the few industries in which no sort of warranties are generally made to customers. In fact, the only two warranties that are commonly made are a 90 day limited express warranty on the behavior of the software and a 90 day warranty on the media. The first warranties that the software will behave substantially in accordance to the provided user manuals; the second warranties that the media the software is provided in will be free of manufacturing defects for 90 days.

No other warranties are made. In fact, the licenses explicitly deny any other warranties, whether express or implied, to the maximum extent permitted by the law.

2.2.10 Liability

Just like software companies try to avoid making any warranties on software, they similarly avoid having any liability (for the promises they didn't make). Most software companies limit their liability to either the cost of the software license or a pre-fixed amount specified in the license. This amount usually isn't very high; within the licenses we studied the highest amount was $500. Some software like Java [6] and WinZip [16], assumed no liability at all.

We want to emphasize that companies limit their liability as much as permitted by the law. They deny being liable, regardless of the cause of damage, regardless of the theory of liability and regardless of whether they have been previously advised of such damages. Their proposed monetary remedy is also the maximum they will pay, even if the remedy fails its essential purpose.

We feel the position software companies take with respect to liability and warranty, leave an open market for companies that want to make more promises to their customers. It could even be a third party company that especializes on setting up software and, in a sense, offering "insurance" against it failing.

2.2.11 Jurisdiction

Since software is sold all around the world, software companies generally make the software users agree to a jurisdiction. Any legal issues that arise are to be resolved according to the laws of that jurisdiction. As expected, all of the software we studied stated as it's jurisdiction the state where the software company resided.

2.2.12 Miscellaneous

In this section we discuss other features that we thought were important, but not generalizable enough to apply to all the licenses.

See Tables 2.2.12, 2.2.12, and 2.2.12 for more details; the features are self-explanatory.

3 Discussion

In this section, we discuss two major licenses in detail. Specifically, we examine the GPL (GNU Public License) [5], an extremely popular open-source license, and we examine the Microsoft EULA (End-User License Agreement), which is distributed with minor variations along with most of its closed-source commercial software. Following these discussions, we present our own opinion regarding the selection of a license. We argue that selectively opening your source code can provide benefits for all parties involved.

	Microsoft Windows XP	Apple OS X
Hardware Permitted	1 computer with at most 2 processors (bundled).	1 Apple-labeled computer.
Network Use	At most 10 concurrent connections to the services provided by the software.	Unlimited concurrent connections to the services provided by the software.
Activation Required	Yes, mandatory within 30 days.	No, not necessary.
Backups Permitted	1 backup copy.	1 backup with copyright notices.
Derivative Work	Cannot reverse engineer, decompile or disassemble.	Cannot reverse engineer, decompile or disassemble.
Resell Policy	One-time permanent transfer.	One-time permanent transfer.
Export Restrictions	Must follow US Export restrictions.	Must follow US Export restrictions.
Warranty	90 days limited express warranty.	90 day limited warranty on media.
Liability	License Fee or $5, whichever is greater.	$50
Jurisdiction	Washington	California
Miscellaneous	Allows installation of 3rd party software. Comes bundled with products that require additional licenses. Allows MS to collect data from your computer in a manner that can't personally identify you.	Need license for non-personal MPEG-2 use.

Table 5: Table summarizing the differences and similarities between the proprietary licenses for selected OSes.

	Microsoft Office	Adobe Photoshop	Macromedia Dreamweaver
Hardware Permitted	1 main computer and 1 portable computer.	1 main computer and 1 portable computer.	1 computer.
Network Use	Alternative network license.	Alternative network license.	Alternative network license.
Activation Required	Yes, after a finite number of product launches.	No, not necessary.	No, not necessary.
Backups Permitted	None. The original media is considered the backup.	Reasonble number.	1 backup with copyright notices.
Produced Work	Cannot create obscene or scandalous work with provided media elements.	Cannot create pornographic or illegal material with stock.	No restriction.
Derivative Work	Cannot reverse engineer, decompile or disassemble.	Cannot reverse engineer, decompile or disassemble.	Cannot reverse engineer, decompile or disassemble.
Resell Policy	One-time permanent transfer.	Permanent transfer.	Permanent transfer.
Export Restrictions	Must follow US Export restrictions.	Must follow US Export restrictions.	Must follow US Export restrictions.
Warranty	90 days limited express warranty.	90 days limited express warranty.	90 days limited express warranty. 90 day limited warranty on media.
Liability	License Fee or $5, whichever is greater.	License Fee.	License Fee or $500, whichever is greater.
Jurisdiction	Washington	California	California
Miscellaneous			Macromedia may audit your use of the software upon reasonable notice.
Comments			Reverse engineering clause is curiously worded; it states that the user may not "reduce the Software to a human perceivable form".

Table 6: Table summarizing the differences and similarities between the proprietary licenses for selected applications.

	Java 1.4.1	Winzip
Hardware Permitted	Unlimited number of computers.	Unlimited number of computers for one (1) person. Or, one (1) computer for an unlimited number of people.
Network Use	Permitted without an additional license.	Alternative Network License.
Activation Required	No.	Within 21-days for unpaid version.
Backups Permitted	[Unlimited number.]	[Unlimited number.]
Produced Work	Developers must publish freely accesible documentation for any software they create that extends the functionality of the Java Platform. Developers are not allowed to create classes under the java, javax or sun packages.	Self-extracting files contain extraction software that may not be altered or modified.
Derivative Work	Cannot reverse engineer, decompile or disassemble.	Cannot reverse engineer, decompile or disassemble.
Resell Policy	Not permitted. Software is non-transferable.	
Export Restrictions	Must follow US Export restrictions.	Must follow US Export restrictions.
Warranty	90 day limited warranty on media.	None.
Liability	$0	$0
Jurisdiction	California	Connecticut
Miscellaneous	Allows Sun to install 3rd party software.	

Table 7: Table summarizing the differences and similarities between the proprietary licenses for selected utilities.

3.1 GPL

Although a large number of open-source software licenses currently exist, the most common is the GNU General Public License (GPL). Created by Richard Stallman, a member of MIT, under the GNU Software Project [4], the GPL license was designed to force software licensed under GPL, and any programs built using the source code, to remain “free”. In essence, GPL allows users the freedom to use, modify, and distribute copies of the original source code. Furthermore, all modified versions of the source code, derivative works, or larger programs containing all or part of the source code must be licensed under GPL and hence remain “free”. GPL, however, does not restrict the right of the software creator to charge for the distribution of the source code. In general, GPL provides a large array of freedom and protection for the rights of software developers. However, because GPL requires all larger software derived or containing parts of source code licensed under GPL to also fall under the GPL license, it is also one of the most restrictive. Hence, commercial software cannot incorporate source code released under GPL. While this restriction is not necessarily bad, conflicts of licensing can prevent software designers from using freely available and open-source code to improve commercial products. Furthermore, an open-source software package may not be combined with source code licensed under GPL if the software package uses an open-source software license that is not compatible with GPL.

3.2 Microsoft EULA

In this section, we discuss in more detail the Microsoft Windows XP software license. We found this license to be the most restrictive of the ones we studied and many of the restrictions seem to be there only to get more money out of the customers. As the analysis shows, other successful software companies have less restrictions and still manage to generate a profit and develop quality software products.

In terms of hardware, the license bundles the software with the computer on which it is installed and limits the number of processors to two. So a user has to keep buying new licenses for the software every time he or she upgrades to a new machine, or obtain multiple licenses if the computer has more than two processors. This makes Windows considerably more costly as an operating system than the alternatives for users who constantly upgrade their machines or who have specialized parallel computers. The license also limits connections to network services provided by the operating system. The user is forced to either pay for a more expensive license to allow for more connections, or to obtain third party software that accomplishes the same purpose. Finally, since the license imposes a one-time permanent transfer policy, users cannot easily obtain used versions of the software at a better market price. They are forced to go back to Microsoft to buy the software.

Other negative aspects about the Microsoft license are that it allows third party software to be installed, requires an activation procedure, and allows Microsoft to collect information from the user. The third party software generally refers to additional software the user wants to run, such as codecs for video formats. However, since Microsoft makes no warranties, the third party software could potentially be dangerous and open a door for malicious programs to attack. Requiring an activation from the user is annoying, intrusive, and forces users to provide information they might not want to share with the company (such as personally identifiable information like address and email). The license also allows Microsoft to collect information from your computer. This information cannot personally identify the user, but what exactly this means is questionable. There is information that doesn’t immediately identify you, but could, with enough research. Since users don’t have the code, they have no way of verifying what information is being transmitted.

In short, we find that the Microsoft license is too restrictive and embodies many of the negative aspects that we feel should not be present in a proprietary license.

3.3 Hybrid Model

In this section, we suggest a novel hybrid licensing model for proprietary software vendors. Under this scheme, portions of the source code are offered as part of the software package, while other aspects are left closed-source. We argue, in particular, that those parts of the program related to security and compatibility should be open-source. This, in conjunction with our numerous editorial remarks regarding specific licenses, constitutes the primary creative contribution of this work.

3.3.1 Why Open?

There are countless arguments for and against making software open-source. We will not reiterate them here, except to highlight those parts of the debate that are most relevant to our conclusions.

One such issue is security. By revealing the source code, some argue, it is possible for hackers or malicious entities to discover vulnerabilities and that this presents a security risk. Of course, this argument misses the point: security through obscurity doesn't work. Software should not be written around the assumption that it is secure so long as the code is not made public. Indeed, such tactics are dangerous and foolish, since a source code leak could render all your security measures moot. A much better tactic is to design your software with actual security measures, using encryption and other accepted techniques. These mechanisms do not depend on the source being a secret; quite the contrary, the mechanism is very public. Instead, the security of the system lies in much smaller and more modular "secrets" in the form of private keys, passwords, and so on. It is not likely that a single leak will cause a major security breach, while this is certainly the case when secrets are hidden in the source code.

Therefore, by allowing the source code related to security to be made public, many people can verify the integrity of your security mechanisms. This will help make your system more secure, and will also instill confidence in your customers by letting them see for themselves that their data will be protected. In our opinion, if a company insists on keeping their source private, claiming that it is a security risk to release it, then that company has a poor understanding of computer security and should probably be avoided.

Another point of contention is compatibility. Proponents of open-source argue that it is easier to write drivers, plugins, and other extensions when they have access to the source. APIs are useful, but are frequently incomplete or lack information related to performance. Exposing the source related to compatibility with hardware and other software allows third-party coders the opportunity to extend and improve the compatibility of your software. This accomplishes something else in addition to increasing the size of your potential user base, which is that it tends to build a community of people who have contributed to the project in some way. This gives people a sense of ownership and they are more likely to continue using your software in the future.

Again, making this portion of your software open-source benefits your customers by increasing compatibility and benefits the vendor by generating a community of contributors.

3.3.2 Why Closed?

This raises the question: why not open the entire code base? For many companies, this is a hair-raising proposition. They may fear that releasing their code will provide people with a disincentive to purchase it, because they can get it for free. Or, they may feel that certain portions of their code contain methods or techniques that they do not yet wish to have made public. We respond to these concerns in order.

Anti-piracy schemes will always be broken. Activation keys will be distributed, key generators will be coded through reverse engineering, and disc images will make their rounds. The point of anti-piracy measures is not to prevent piracy, because that would be both impossible and would almost necessarily inconvenience paying customers. Instead, the goal of anti-piracy measures is to make it *unprofitable* to pirate. That is, the measures should be costly enough to circumvent that *most* people won't bother to pirate it[3]. There will always be people who will make it their mission in life to break so-called DRM schemes. But most people aren't malicious, and they aren't thieves. But if they don't feel your software is worth the price, they won't buy it. And some of *those people*, the ones who wouldn't buy the software anyway, will reason that you don't lose any money if they pirate it.

The reasoning for keeping a portion of your software proprietary follows from these observations. If most of your code is available, anyway, the amount of profit from breaking open the remainder of the code is decreased! This is especially powerful when the part of your code that is open-source compiles into a functional, though incomplete, system[4]. It is worth noting that releasing something without source code can be considered a form of anti-piracy measure, in that people assume that it is impossible to recover code from a binary. This is a fallacy; it is possible to reverse engineer a good portion of the code functionality given a running program. Decompilers and a good knowledge of programming techniques are often enough to determine how something was coded. After all, binaries are really just code in a human-obfuscated language. This isn't to say that you get a perfect replica of the original code, and the code won't be commented (if it ever was), but if you are trying to hide an algorithm or security measure, you will fail. Again, it is a question of efficiency. It is time-consuming to reverse engineer something, and if your software is cheap enough, or enough of it is freely available, the vast majority of people won't find the exercise worthwhile.

Still, keeping a small portion of the code proprietary will at least discourage the casual user from breaking it apart. That, and the mere fact that companies will have the illusion that something is being withheld from the customer may appeal to certain companies, thus making this scheme more appealing.

3.3.3 Compromise

All of this suggests a compromise: open a large, functional portion of your code, while keeping a small, select portion proprietary. Opening the security and compatibility portions of your code serves to, among other things:

1. Increase security through independent auditing.
2. Cultivate a contributing community.
3. Improve compatibility.

Meanwhile, keeping a portion of the code closed has an appeal to companies because it:

1. Discourages casual piracy.
2. Temporarily obfuscates algorithms or security measures.
3. Appeals to companies.

4 License Selection Engine

As part of our effort to help the average programmer choose an appropriate license for their software, we have created the License Selection Engine [12] that suggests licenses based upon their responses to

[3]Another way to make circumvention unprofitable is, of course, to lower the price of your software. If piracy is rampant, a smart company will consider this as a possible recourse. The RIAA, for example, hasn't quite figured this out.

[4]For example, Apple releases the Darwin OS as free open-source software. It is completely functional, but lacks, among other things, the proprietary Aqua GUI. These additional components turn Darwin into Mac OS X.

a series of questions. Currently, there are a wide variety of licenses available and the choice of license for a particular piece of software can have far reaching consequences for the amount of compensation or recognition that a programmer or firm can receive for producing a particular piece of software. Since most small software companies or individual programmers are not skilled in the complex legal matters surrounding software licenses, we have attempted to aid in the choice of a license by providing a variety of suggestions and pointers towards more detailed information about each license once a suggestion has been made.

The License Selection Engine is available in a web-based format and covers a wide variety of licenses ranging from propriety Microsoft EULA [11] to the GNU public license [4]. The programmer enters the website and is presented with a series of questions determining the broad category of license that is appropriate; this is shown in Figure 1. This involves asking whether the software is open source and whether the programmer plans to charge for the program. As Figure 2 illustrates, once the broad category is established, more detailed questions are asked about the types of rights that the programmer wants to hold over the software, and the list of possible licenses is narrowed. At the end of answering around 10 questions, the license selection engine compares the survey results with a matrix of licenses and their properties and makes a suggestion based upon this correlation. See Figure 3 for an example. Because of the breadth of licenses available, there are licenses that cover almost every possible survey response.

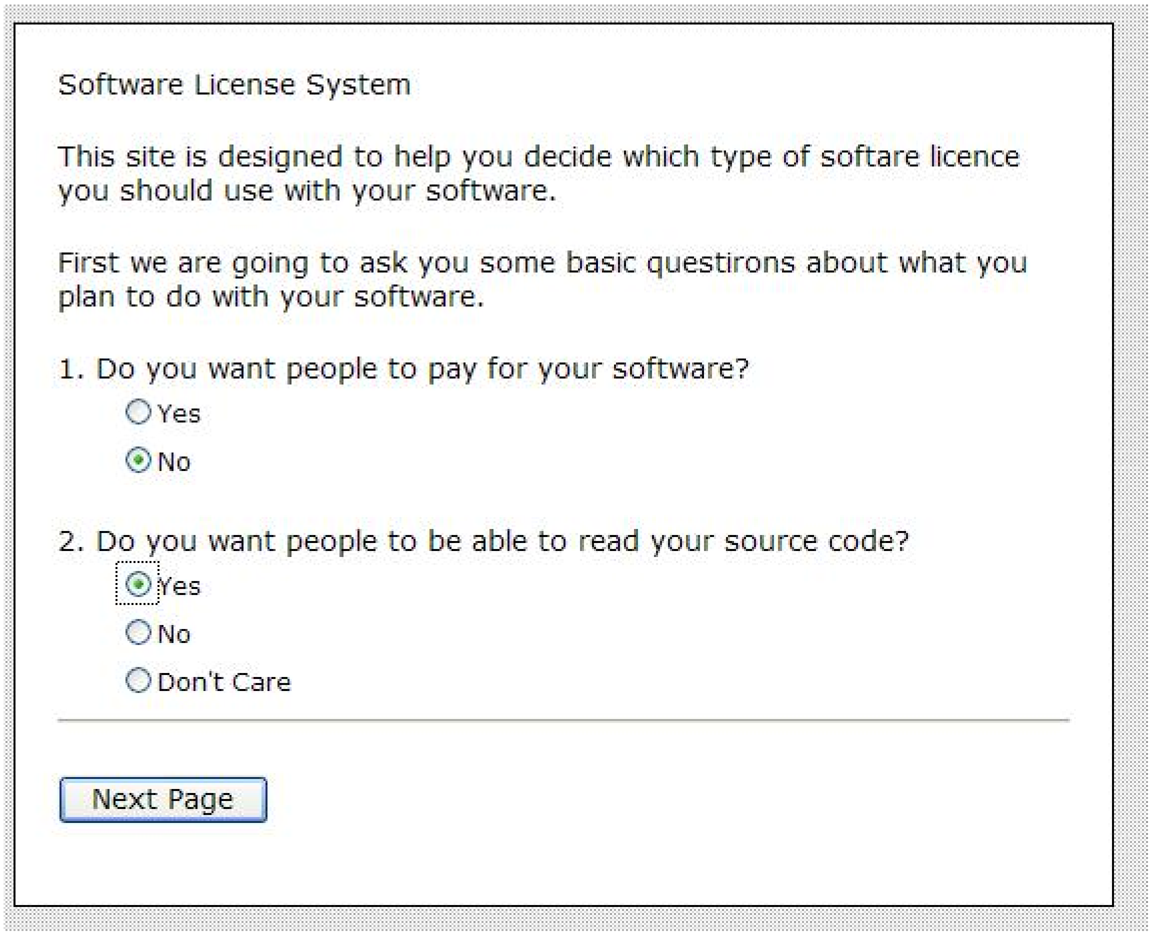
In Section 2, we gave tables showing the various rights provided by each licence. The license selection engine consults these tables after accumulating the results of the survey in order to suggest an appropriate license.

4.1 Technical Details

The License Selection Engine is implemented using a perl script that can run on any web server that supports perl extensions. The questions and matrix for licenses are stored in text files that can easily be modified by a user. The web server used to run the engine in this case is a standard Red Hat Linux box that is running Apache 2.0 and Perl 5. The data that is used to run the engine is stored into separate database files. These are essentially text files that are in a human-readable format for easy editing by a user who wishes to extend the system. There is one file, `questions.txt`, which contains the questions to ask the user and which actions to perform if a given question is answered positively. For example, on the first page the question about open versus closed source contains a pointer to the appropriate next page of questions. The other file, called `licenses.txt`, contains each license covered by the License Selection Engine, along with a short description, and the matrix entry of how a respondent should answer the questions in order to have the particular license selected. The modularity of the system allows the engine to be easily updated to take advantage of new licenses or make modifications to reflect changes in the wording of existing licenses.

4.2 Related Work

There are more primitive versions of the License Selection Engine on the web [13, 14], but most cover few licenses and we have found none that covers both open and closed source programs. Also, given the extendable nature of the License Selection Engine, we anticipate that the number of licenses covered by our system can grow indefinitely. The ultimate in license selection would be able to generate a perfect custom license on the fly, in a similar way to how the Creative Commons [2] site generates license for various types of electronic media. This a more difficult task for software licenses since there are a very large number of rights that can be applied in a variety of different situations.



Software License System

This site is designed to help you decide which type of softare licence you should use with your software.

First we are going to ask you some basic questirons about what you plan to do with your software.

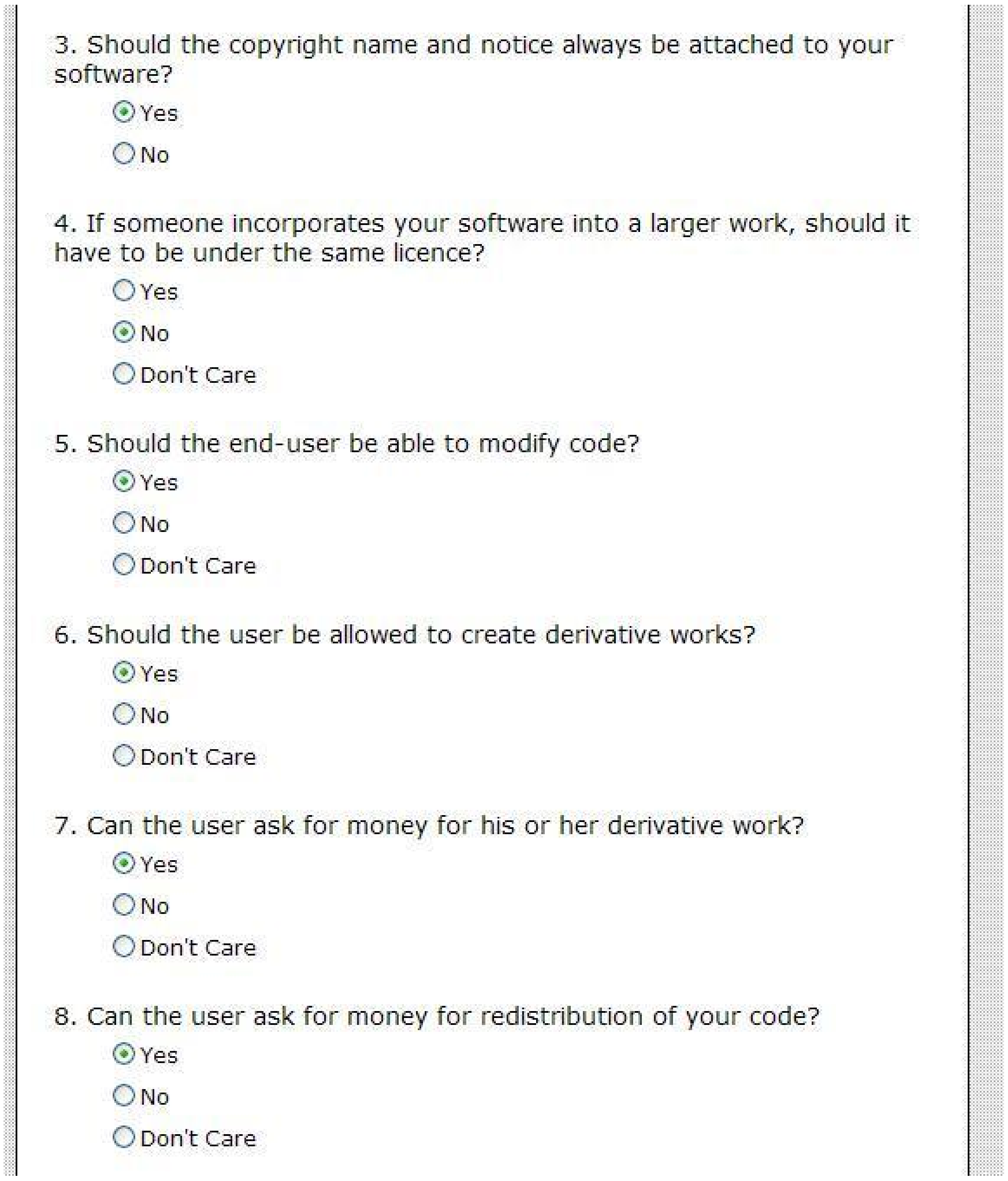
1. Do you want people to pay for your software?

Yes [✓ No]

2. Do you want people to be able to read your source code?

[✓ Yes] No Don't Care

Figure 1: The first few questions determine the broad category of license that will be used.



3. Should the copyright name and notice always be attached to your software?

[✓ Yes]
No

4. If someone incorporates your software into a larger work, should it have to be under the same licence?

Yes
[✓ No]
Don't Care

5. Should the end-user be able to modify code?

[✓ Yes]
No
Don't Care

6. Should the user be allowed to create derivative works?

[✓ Yes]
No
Don't Care

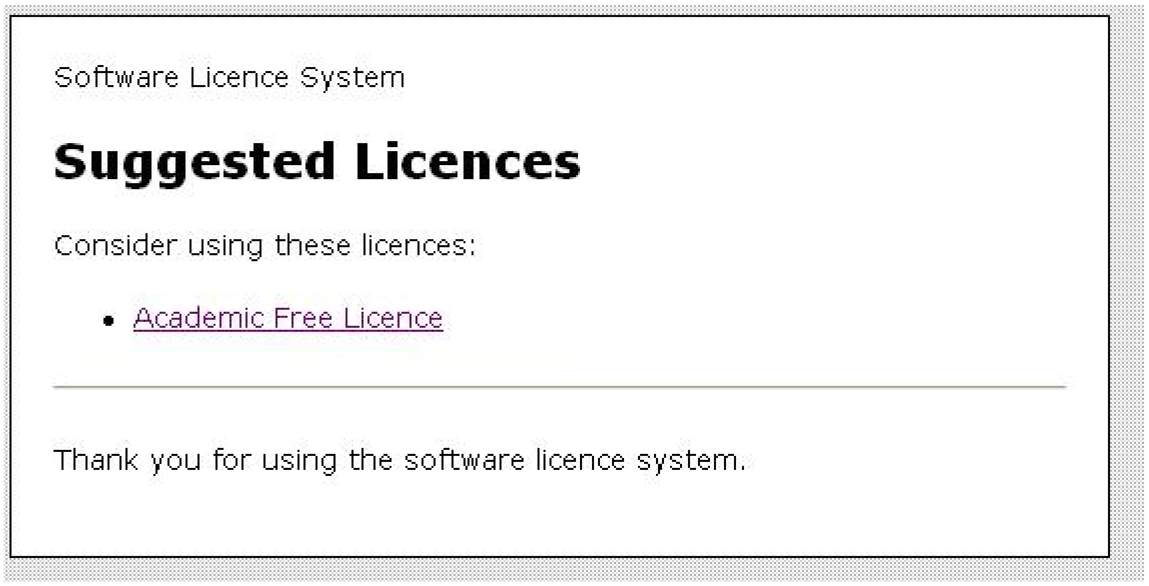
7. Can the user ask for money for his or her derivative work?

[✓ Yes]
No
Don't Care

8. Can the user ask for money for redistribution of your code?

[✓ Yes]
No
Don't Care

Figure 2: More in depth questions allow the system to narrow down the field of possible licenses.


Software Licence System

Suggested Licences

Consider using these licences:

- Academic Free Licence

Thank you for using the software licence system.

Figure 3: In this case, the License Selection Engine has found one license (the academic free license). You can click the link to find out more information about the license and the text that should be included with the software.

5 Contributions

This paper has made the following contributions:

1. Presented taxonomy of software licenses, both open and closed-source.
2. Examined two major licenses in detail.
3. Implemented a software License Selection Engine.
4. Argued in favor of partially opening source code in the context of proprietary software.

In particular, it is worth noting the original, creative contributions of this work. We have presented a more complete taxonomy than has ever before been attempted, and provided a License Selection Engine to present this material in an intuitive and useful way. We provided our own opinions regarding what licenses are worthwhile, and which we feel are largely flawed. Finally, and most importantly, we discussed a unique hybrid licensing model of our own design.

6 Acknowledgments

We would like to thank Professor Robert Rines and Ethel Machi for their invaluable instruction and input.

References

[1] Licensing. *Adobe Photoshop*
URL http://www.adobe.com/support/downloads/license.html

[2] Creative Commons.
URL http://creativecommons.org/

[3] Copyleft.
URL http://www.gnu.org/copyleft/copyleft.html

[4] GNU Operating System - Free Software Foundation.
URL http://www.gnu.org/

[5] Licensing. *GPL*
URL http://www.gnu.org/copyleft/gpl.html

[6] Licensing. *Java*
URL http://java.sun.com/j2se/1.4.2/j2sdk-1_4_2_04-license.txt

[7] Licensing. *Open Source Initiative OSI*
URL http://www.opensource.org/licenses/ (visited 2004, April 11).

[8] Licensing. *Mac OS X*
URL http://store.apple.com/Catalog/US/Images/singleuser.html

[9] Licensing. *Macromedia*
URL http://www.macromedia.com/software/dreamweaver/productinfo/faq/

[10] Licensing. *Microsoft Office*
URL http://www.microsoft.com/office/eula/en.mspx

[11] Licensing. *Microsoft Windows XP Professional*
URL http://proprietary.clendons.co.nz/licenses/eula/windowsxpprofessional-eula.htm

[12] E. Nordlander, D. Loreto, A. Oliner, R. Woo. *License Selection Engine.*
URL http://www.nordlander.com/6.931/lic.pl

[13] Open Source License Quick Reference Chart.
URL http://pgl.yoyo.org/lqr/ (visited 2004, April 11).

[14] Quick Reference For Choosing a Free Software License.
URL http://zooko.com/license_quick_ref.html (visited 2004, April 11).

[15] Various Licenses and Comments about Them. *GNU Project - Free Software Foundation (FSF)*
URL http://www.fsf.org/licenses/license-list.html (visited 2004, April 11).

[16] Licensing. *Winzip*
URL http://www.winzip.com/license.htm