Open Source Licensing, Contract, and Copyright Law

1.1

Under the laws of the United States and European countries, copyright is automatically granted to original expressions of ideas, whether they are in the form of text, sounds, or imagery. This protection applies to various creative works, including writings, music, art, and more. The copyright holder has exclusive rights to reproduce, distribute, and create derivative works based on their original expression. Copyright protection lasts for a specific duration, typically the life of the creator plus several years. During this time, others are not allowed to copy or commercially exploit the copyrighted work without permission. However, copyright does not protect the underlying idea itself, only its specific expression. Certain limitations, such as fair use, allow for limited use of copyrighted material for purposes such as commentary, criticism, reporting, or teaching. After the copyright term expires, the work enters the public domain and can be freely used and exploited by anyone.

1.2

In the United States, creators automatically obtain the rights to their work upon its completion in a fixed medium. No registration or signed writing is required to preserve these rights, as they are granted by the operation of statutory law. However, creators often rely on others, such as publishers, to produce and distribute their works on a large scale. The relationship between creators and publishers can be uneasy, as both parties seek control over the work and the income derived from its commercial exploitation. Publishers, due to the capital-intensive nature of the publishing industry, typically have more leverage in these negotiations. Licensing agreements, in which creators receive royalties in exchange for the use of their work, are common between creators and publishers. However, in the software publishing industry, royalties to individuals are less prevalent because software development often involves large teams and substantial capital investment, resulting in the work being considered "work for hire" owned by the employer. While creators hold a legal monopoly over their works, the substantial costs of development and distribution often lead to corporations benefiting more from copyright than the individual creators themselves.

1.3

Open source licensing emerged as a response to the traditional distributor-driven model of copyright licensing. The primary purpose of open source licensing is to prevent exclusive exploitation of a work. In the traditional model, publishers, driven by the desire for income from the exploitation of the work, retain exclusive rights and are reluctant to share the value of the work with others.

Unlike artistic or aesthetic works, which rely on their original form of expression, software is both functional and dynamic. Each program has both its intended purpose and the potential to be used for other functions. When a consumer purchases commercial software, they acquire the right to use it for its intended purpose, but they are restricted from copying or creating derivative works. Open source licensing removes these restrictions, allowing users to freely distribute and modify the software, as long as they maintain the same open source licensing terms.

Open source licensing brings several benefits over traditional proprietary licensing models. First, it fosters innovation by enabling a large number of programmers to contribute to a work, increasing its value. Second, it enhances reliability as more programmers can debug and improve the software, and knowledgeable users

can address limitations and bugs. Third, it ensures longevity as open source software can be revived, adapted, or rewritten by subsequent users, even after commercial software becomes obsolete.

Open source licensing encourages collaboration, community-driven development, and the sharing of knowledge and resources. It challenges the traditional distribution model and promotes the principles of openness and accessibility in software development and distribution.

1.4

The licensing of open source software involves two essential parts: the assertion of the licensor's rights to license the work and the grant of rights to the licensee. The licensor must have the necessary rights to grant to the licensee for the license to be effective. However, complications arise when a third party claims legitimate legal rights to the licensed work.

In the case of copyrights, a creator can confidently license their original work, assuming they have not plagiarized it. Copyright protection is granted automatically upon creation, without the need for filing or formal notice. On the other hand, obtaining and retaining a patent is more complex. It requires filing extensive paperwork, including an explanation of the novelty of the patent and how it differs from existing processes or mechanisms. Patent holders are vigilant in protecting their rights and may engage in litigation to extend their patent boundaries and restrict others' patents.

Patent licensing in the context of open source software presents challenges. A licensor may not have the inclination or resources to defend a licensed patent, which could have negative consequences for licensees. If a rival patent holder successfully narrows or eliminates a licensed patent through litigation, the licensee may be liable for infringement due to continued use of the patent. In such cases, licensees may need to take action to protect the licensor's patent, even if it involves costly litigation.

Another problem is the possibility of undisclosed patent claims that apply to the licensed software. Neither the licensor nor the licensee may be aware of these claims. Since licensors can only license works that belong to them, the existence of a software license does not protect the licensee from infringement claims by third-party patent holders. This issue is particularly challenging considering the frequent granting of software patents, which can be vague in their scope.

Overall, navigating patent issues in open source licensing can be complex and require careful consideration. The potential for undisclosed patent claims and the lack of resources or willingness to defend patents by licensors can pose risks to licensees.

1.5

The Open Source Definition, as defined by the Open Source Initiative (OSI), outlines the criteria that licenses must meet to be considered "Open Source." It serves as a guideline for determining whether a license qualifies as open source, and the OSI also certifies licenses that align with this definition.

The Open Source Definition consists of the following criteria:

Free Redistribution: The license should not restrict the selling or giving away of the software as part of an aggregate software distribution. No royalties or fees should be required for such distribution.

Source Code: The program must include source code, and the distribution should allow both the source code and the compiled form to be distributed. If the product is distributed without source code, there should be a

well-publicized means of obtaining the source code at a reasonable cost or for free.

Derived Works: The license must allow modifications and derived works to be created and distributed under the same terms as the original software license.

Integrity of The Author's Source Code: If modifications to the source code are allowed, the license may require the distribution of patch files along with the modified source code. The license should explicitly permit the distribution of software built from modified source code, and it may require that derived works have a different name or version number.

No Discrimination Against Persons or Groups: The license must not discriminate against any individual or group of individuals.

No Discrimination Against Fields of Endeavor: The license must not restrict anyone from using the software in specific fields of endeavor.

Distribution of License: The rights granted by the license should apply to all recipients of the software without the need for additional licenses.

License Must Not Be Specific to a Product: The rights granted by the license should not be dependent on the software being part of a particular software distribution. If the software is extracted from the distribution and used or distributed under the terms of the license, the same rights should apply.

The License Must Not Restrict Other Software: The license must not impose restrictions on other software distributed alongside the licensed software.

The License Must Be Technology-Neutral: The license should not be dependent on any specific technology or style of interface.

These criteria ensure that open source licenses promote open distribution, access to source code, the freedom to modify and distribute derived works, and non-discrimination. By adhering to these principles, open source licenses foster innovation, collaboration, and the sharing of software.

1.6

Warranty disclaimers are commonly included in open source licenses, although they are not a requirement for a license to be considered open source. To understand the effect of warranty disclaimers, it's important to have a basic understanding of warranties and their implications.

There are different types of warranties associated with the acquisition of a product. An express warranty is a specific guarantee made by the seller to the buyer regarding the performance or characteristics of the item being sold. If the product does not meet the stated warranty, the buyer has remedies such as returning the product or receiving a functioning replacement.

A warranty of merchantability is an implied warranty that applies to goods sold by merchants. It guarantees that the goods are fit for their intended use. For example, if you buy rope from a hardware store, there is an implied warranty that the rope will function as rope typically does. This type of warranty is generally applicable only to merchants.

A warranty of fitness for a particular purpose is another type of implied warranty. It applies when the buyer relies on the seller's expertise or advice regarding the suitability of the product for a specific purpose. For

example, if you inform the seller that you intend to use the rope to pull a car out of a ditch, and the seller assures you that the rope is suitable for that purpose, there is an implied warranty of fitness for that particular purpose.

A warranty against infringement is unique to intellectual property. It is a guarantee by the seller that the work being sold does not infringe on the intellectual property rights of others.

Warranty disclaimers in open source licenses aim to limit the liability of the licensor. They typically state that the software is provided "as is" and without any warranties. This disclaimer helps protect the licensor from potential claims related to the performance, functionality, or infringement of the software. However, it's important to note that warranty disclaimers may not be foolproof, as contrary representations or agreements made as part of a sale could nullify the disclaimer and result in liability.

It's advisable for licensors to consult with an experienced lawyer to ensure that the warranty disclaimers in their licenses are enforceable and comply with relevant state or federal laws. Additionally, warranty disclaimers in open source licenses do not preclude the possibility of developers signing separate support contracts to provide additional services or warranties for the software.

The MIT, BSD, Apache, and Academic Free Licenses

2.1

The MIT License, which is one of the simplest licenses discussed in the book, grants permissions to users to deal with the software without restrictions. It begins by stating the copyright information, including the year of publication and the copyright holders.

The license grants the following rights to any person obtaining a copy of the software and associated documentation files (the "Software"):

The right to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software. The right to permit others to do the same, subject to the conditions of the license. The first condition of the license requires that the copyright notice and permission notice are included in all copies or substantial portions of the Software. This ensures that future users of the software are aware of the copyright and licensing terms.

The second condition of the license includes a warranty disclaimer, which states that the software is provided "as is" without any warranties, express or implied. It disclaims warranties of merchantability, fitness for a particular purpose, and non-infringement. It also states that the authors or copyright holders shall not be liable for any claims, damages, or other liability arising from the use of the software.

The warranty disclaimer is an important aspect of the license as it helps protect the authors or copyright holders from potential liability associated with the software.

It's worth noting that the MIT License is a permissive open source license, allowing users to freely use, modify, and distribute the software with limited restrictions.

2.2

The BSD License, specifically the UCB/LBL form, is another open source license that is slightly more restrictive than the MIT License. It starts with the same copyright notice as the MIT License:

Copyright (c) , All rights reserved.

Similar to the MIT License, the BSD License permits redistribution and use of the software in both source and binary forms, with or without modification, under certain conditions.

One notable difference in the BSD License is the provision regarding advertising materials. Prior to 1999, the license required that advertising materials mentioning the software must acknowledge the University of California, Lawrence Berkeley Laboratory as the developer of the software. However, this clause was removed in the 1999 amendment of the license.

The remaining conditions of the BSD License are similar to those of the MIT License, including the retention of copyright notice, conditions, and disclaimer in both source code and binary redistributions. There is also a provision that prohibits the use of the creator's name or organization's name to endorse or promote products derived from the software without prior written permission.

Finally, the BSD License includes a disclaimer of warranties, similar to the MIT License, stating that the software is provided "as is" without any warranties, and the licensors or contributors shall not be liable for any damages arising from the use of the software.

Overall, the BSD License is a permissive open source license that allows for the free use and redistribution of the software, with some additional conditions compared to the MIT License.

2.3

The Apache License, version 1.1, is another open source license that is slightly longer than the licenses discussed earlier but operates in a similar way. It starts with the copyright notice:

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Similar to the BSD License, the Apache License permits redistribution and use of the software in both source and binary forms, with or without modification, under certain conditions.

The first two conditions are identical to the BSD License, requiring the retention of copyright notice, conditions, and disclaimer in both source code and binary redistributions.

The Apache License includes an additional condition regarding the acknowledgment of the creator's contribution to the work. It states that the end-user documentation included with the redistribution must include an acknowledgment that the product includes software developed by the Apache Software Foundation. Alternatively, the acknowledgment may appear in the software itself, following standard practices.

The license also includes a provision that the names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from the software without prior written permission.

Another provision of the Apache License states that products derived from the software may not be called "Apache" or include "Apache" in their name without prior written permission from the Apache Software Foundation. This provision aims to prevent potentially damaging associations between the creator and derivative works.

The Apache License includes a warranty disclaimer similar to the previous licenses, stating that the software is provided "as is" and the Apache Software Foundation and its contributors shall not be liable for any damages arising from the use of the software.

The license concludes with clauses acknowledging the contributions of individuals and public domain software upon which the distributed software is based. These clauses do not impose any obligations on the user but serve to give credit and recognition to the contributors.

Overall, the Apache License, version 1.1, is a permissive open source license that allows for the free use, modification, and redistribution of the software, with certain conditions and acknowledgments.

2.4

The Apache License, version 2.0, released in January 2004, is a thorough revision of the Apache License. It is more detailed and specific in outlining the rights granted compared to the previous version (v1.1). The key differences in v2.0 include explicit provisions for patent rights granted by the license, the use of other licenses for derivative works, and the concept of "Contributions" to the licensed work.

The license begins with definitions:

"License" refers to the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of the document. "Licensor" refers to the copyright owner or entity authorized by the copyright owner to grant the License. "Legal Entity" refers to the union of the acting entity and other entities that are controlled by or have control over the acting entity. "You" (or "Your") refers to an individual or Legal Entity exercising the permissions granted by the License. "Source" form refers to the preferred form for making modifications, including source code, documentation source, and configuration files. "Object" form refers to any form resulting from mechanical transformation or translation of a Source form, such as compiled object code or generated documentation. "Work" refers to the work of authorship, in Source or Object form, made available under the License. "Derivative Works" refers to works, in Source or Object form, based on or derived from the Work, representing an original work of authorship. "Contribution" refers to any work of authorship, including the original version of the Work and any modifications or additions to it, intentionally submitted to the Licensor for inclusion in the Work. "Contributor" refers to the Licensor and any individual or Legal Entity that has submitted a Contribution. The grant of copyright license in Section 2 states that each Contributor grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

Section 3 addresses the grant of a patent license, stating that each Contributor grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work. This license applies only to patent claims licensable by the Contributor that are necessarily infringed by their Contribution(s) alone or in combination with the Work. However, if You initiate patent litigation against any entity alleging patent infringement by the Work or a Contribution incorporated within the Work, your patent licenses granted under the License for that Work shall terminate.

Section 4 discusses redistribution, stating that You may reproduce and distribute copies of the Work or Derivative Works, with or without modifications, in any medium, provided certain conditions are met. These conditions include giving recipients a copy of the License, including prominent notices if any files are

modified, retaining copyright, patent, trademark, and attribution notices, and including readable copies of the attribution notices from the NOTICE file if applicable.

Section 5 clarifies that any Contribution intentionally submitted by You to the Licensor shall be governed by the terms and conditions of the License, unless otherwise explicitly stated. Separate license agreements regarding Contributions may exist and should not be superseded or modified by the License.

Section 6 prohibits the use of the Licensor's trade names, trademarks, service marks, or product names except for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

Sections 7 and 8 disclaim warranties and limit liabilities, stating that the Work and Contributions are provided "as is" without warranties or conditions, and Contributors shall not be liable for any damages, except as required by applicable law.

Section 9 allows You, during redistribution of the Work or Derivative Works, to offer and charge a fee for acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with the License. However, You may act only on Your own behalf and on Your sole responsibility, and You must agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by them as a result of Your acceptance of such obligations.

The Apache License, version 2.0, provides clearer pathways for open and non-open development, allowing licensees to choose whether to make Contributions and how to license their Derivative Works. It explicitly permits licensing Derivative Works under different licenses, as long as the conditions of the Apache License, v2.0, are met.

2.5

The Academic Free License (AFL) is a software license that is similar to the Apache License, v1.1. The AFL includes provisions that forbid claims of endorsement by the work's creator, require attribution to the creator, disclaim warranties, and permit the distribution of the original work and derivative works with certain limitations.

In addition to the provisions found in the Apache License, the AFL adds four more provisions. Two of these provisions relate to patent law, allowing the open source use of patents owned or controlled by the licensor. The other two provisions govern the choice of law and the shifting of attorney fees.

The AFL grants a copyright license that allows the licensee to reproduce the original work, prepare derivative works based on it, distribute copies of the original work and derivative works, perform the original work publicly, and display the original work publicly.

The license also grants a patent license that allows the licensee to make, use, sell, and offer for sale the original work and derivative works, under the patent claims owned or controlled by the licensor that are embodied in the original work.

The AFL requires the licensor to make the source code of the licensed program available in some form, along with documentation on how to modify the original work.

The license includes provisions that exclude the use of the licensor's name or trademarks for endorsement or promotion without express written permission. It also includes provisions on attribution rights, warranty of

provenance, limitation of liability, termination for patent action, jurisdiction and governing law, attorney fees, and miscellaneous provisions.

The license is the complete agreement concerning the subject matter and includes definitions of "You" as the licensee and grants the right to use the original work in ways not otherwise restricted by the license or by law, with the licensor promising not to interfere with or be responsible for such uses.

The license is copyrighted and may be copied and distributed without modification, but any modifications to the license require the express written permission of the copyright owner.

2.6

The BSD, MIT, and Apache licenses have been widely used in both open source and commercial software applications. These licenses have contributed to the widespread adoption of the programs they license, often through incorporation into proprietary products. Examples include BSD Unix becoming the basis for commercial versions of Unix and the TCP/IP software stack being used in commercial TCP/IP stacks.

While commercial versions of these programs have been successful, they have not discouraged open source development. Open source versions of these programs have continued to be widely available for modifications and improvements. For example, Berkeley Unix, despite setbacks from a lawsuit, still has millions of installations and continues to be modified and improved. Other parts of Berkeley Unix, such as BIND and Sendmail, have also thrived in the open source ecosystem.

These licenses, such as the BSD and MIT licenses, are ideal for situations where wide deployment of ideas is desired, regardless of whether it results in open source or proprietary software. The success of commercial developments based on programs distributed under these licenses does not undermine the purpose of open source licensing. The licenses were designed to allow for commercial use and were foreseen by their drafters.

While some may argue that the widespread adoption of commercial versions discourages open source development, it is not inconsistent with the licenses' text and intent. The original developers of BSD and the X Window System intended for their software to be used in this way and even built their own companies based on the open source software they created.

Additionally, for certain types of programs, additional license restrictions may not be necessary to maintain an open development model. The Apache license, for example, has retained its dominant market share due to strong branding and standards compliance. The license's requirement that derived works cannot use the Apache name provides protection, and adherence to standards prevents proprietary extensions.

These licenses allow forking and project fragmentation, as seen in the multiple versions of BSD. However, this is more a result of complex social dynamics in large software projects rather than the licenses themselves. The licenses have provided a foundation for both open source and commercial success, allowing for wide adoption and continued development of the software.

The GPL, LGPL, and Mozilla Licenses

3.1

The GNU General Public License (GPL) is a foundational open source license created by the Free Software Foundation (FSF). It is the preferred license for projects authorized by the FSF, including GNU Emacs Editor, GNU C Compiler, and the GNU/Linux kernel. The GPL is designed to guarantee freedom to share and modify free software.

The preamble of the GPL explains its intentions and premise, emphasizing the importance of freedom rather than price. It ensures that users have the freedom to distribute copies of the software, access the source code, make modifications, and know their rights. The GPL also addresses the issue of software patents, requiring that any patents related to the licensed software must be licensed for everyone's free use.

The GPL contains specific provisions that outline its scope and limitations. It allows the distribution of verbatim copies of the program's source code, provided that copyright notices and disclaimers are included. The license permits charging a fee for the physical act of transferring copies and offering warranty protection in exchange for a fee.

The GPL's most important aspect is the concept of copyleft. When a derivative work is created based on the GPL-licensed program, it must also be licensed under the GPL and subject to its restrictions. This ensures that derivative works remain free and open source.

The GPL requires modified versions of the program to include prominent notices of changes, license those modifications under the GPL, and inform users of their rights. It also clarifies that certain sections of code may be distributed separately and not subject to the GPL if they are considered independent works.

To comply with the GPL, the licensees must make the source code available either by including it with the program or by providing a written offer to distribute the source code. The GPL defines source code as the preferred form of the work for making modifications, including all associated interface definition files and compilation scripts. There are options for noncommercial distribution as well.

Overall, the GPL ensures that free software remains free and that users have the freedom to access, modify, and distribute it. It imposes restrictions on derivative works to maintain openness and prevent proprietary restrictions.

Aa detailed explanation of several key provisions found within the GNU General Public License (GPL), a widely used open source software license. Here is a summary of the main points discussed:

Distribution of Source Code: The GPL requires that if executable or object code is made accessible for copying, the corresponding source code must also be made available from the same location. This ensures compliance with the GPL's requirements.

Violation Consequences: If a licensee violates the terms of the GPL, such as by distributing a proprietary derivative work based on GPL-licensed code, their rights under the license are automatically terminated. This can result in legal liability, including injunctions and damages, for the ex-licensee.

License Termination: Licensees who have received copies or rights under the GPL from a licensee in compliance with the license will not have their licenses terminated as long as they remain in full compliance.

Acceptance of the License: Licensees are not compelled to accept the GPL, but if they choose to modify or distribute the program or works based on it, they are indicating their acceptance of the license and its terms.

Contractual Privity: Section 6 of the GPL establishes a relationship between the original licensor and all licensees of the work, regardless of the number of distributions. This helps ensure the standing of the licensor

to bring a lawsuit against any licensee in case of violations.

Adding Restrictions: Licensees are prohibited from imposing additional restrictions on the exercise of rights granted under the GPL. This provision aims to maintain the openness and freedom associated with the GPL.

Liability for Third Parties: Licensees are not responsible for enforcing compliance by third parties to the GPL. This provision protects innocent distributors from liability for violations committed by others.

Limitations on Altering the License: The GPL states that no outside act, including court judgments or patent claims, can limit or modify the terms of the license. If such conditions contradict the GPL, licensees may not distribute the program.

Compatibility with Other Licenses: Section 10 addresses the compatibility of the GPL with other licenses. Licensees seeking to incorporate GPL-licensed parts into other free programs with different distribution conditions should seek permission from the original author.

Disclaimers of Warranty and Liability: The GPL includes disclaimers of warranty and limits liability for damages arising from the use or inability to use the program, except where required by applicable law or agreed to in writing.

3.2

The GNU Lesser General Public License (LGPL) is a license created by the Free Software Foundation (FSF) to allow certain programs, specifically subroutine libraries, to be licensed under an FSF license while permitting them to link with non-GPL software. The LGPL addresses the issue of linking libraries with non-free programs, which could potentially violate the GPL.

The LGPL allows libraries to be linked with non-GPL licensed programs, including proprietary software. However, the LGPL suggests that libraries should be licensed under the GPL in certain circumstances. It provides an alternative license that preserves many benefits of the GPL for libraries while allowing greater flexibility in terms of linking with non-GPL software.

The LGPL defines a library as a collection of software functions and/or data that can be conveniently linked with application programs to form executables. It distinguishes between a "work based on the Library," which is subject to the LGPL's restrictions, and a "work that uses the Library," which falls outside the scope of the license.

The LGPL allows licensees to copy, distribute, and modify the library's source code under certain conditions, including prominently displaying copyright notices and disclaimers of warranty. Licensees must ensure that modified versions of the library are themselves software libraries and distribute them under the LGPL.

If a program uses a facility provided by the library, licensees must make a good faith effort to ensure that the facility can still operate even if the program does not supply the necessary function or table. This provision aims to maximize the library's portability and independent functionality.

Licensees have the option to apply the terms of the ordinary GNU General Public License (GPL) to a copy of the library instead of the LGPL. However, this change is irreversible for that copy, and subsequent copies and derivative works would fall under the GPL.

The LGPL requires the distribution of the complete corresponding machine-readable source code when distributing the library in object code or executable form. If object code distribution is offered from a

designated place, the equivalent access to the source code must also be provided.

The LGPL excludes from its scope the "work that uses the Library" in isolation, considering it not a derivative work of the library. However, when a program that uses the library is linked with the library, the resulting executable is considered a derivative work and is subject to the terms of the LGPL.

The LGPL strikes a balance between allowing flexibility for programs that use LGPL-licensed libraries and imposing restrictions when distributing the program together with the library. It provides an alternative licensing option for libraries while ensuring compatibility with non-GPL software.

The excerpt describes various provisions and distinctions within the LGPL. Here's a summary of the key points:

Different types of work: The LGPL distinguishes between the LGPL-licensed Library, the "work that uses the Library," and the combined "work that uses the Library" and Library together.

Derivative work: If a work incorporates a header file from the Library or relies on certain limited aspects of the Library, it may be considered a derivative work of the Library. The threshold for this determination is not precisely defined.

Distribution requirements: The LGPL allows the "work that uses the Library" to be distributed under any license, while the Library itself must be distributed under the terms of the LGPL. Executables containing the combined work must also follow the LGPL.

Distribution options: The distributor can choose to distribute the complete corresponding source code for the Library and the object/source code of the "work that uses the Library." Alternatively, they can use a suitable shared library mechanism for linking.

Notice and attribution: Distributors must provide prominent notice that the Library is used in the work and that it is covered by the LGPL. Copyright notices for the Library must also be included.

Distribution limitations: The LGPL does not permit distributing a combined work if it includes proprietary libraries or components that are not covered by the LGPL.

Section 6 and Section 7: Section 6 addresses distributing a "work that uses the Library" along with the Library. Section 7 allows placing a work based on the Library side-by-side with other library facilities under different licenses, subject to certain conditions.

Warranty and liability: The LGPL provides no warranty for the Library, and the copyright holders and other parties cannot be held liable for damages.

How to apply the terms: The excerpt concludes with instructions on how to apply the LGPL to new libraries, including attaching notices and providing contact information.

3.3

The Mozilla Public License 1.1 (MPL 1.1) is a software license that was developed by Netscape Communications and released in 1998. It is a hybrid license that incorporates elements of the GNU General Public License (GPL) and the BSD licenses.

The MPL 1.1 allows for the distribution and modification of covered code, which includes both the original code and any modifications made by contributors. It grants users a world-wide, royalty-free, non-exclusive

license to use, reproduce, modify, display, perform, sublicense, and distribute the covered code. The license applies to both source code and executable versions of the software.

Under the MPL 1.1, the source code of the covered code must be made available in source code form, either on the same media as the executable version or through an accepted Electronic Distribution Mechanism. The source code must remain available for a specified period of time.

The license requires contributors to document their modifications and provide attribution to the original developer. It also addresses intellectual property matters, including third-party claims and patent licenses.

The MPL 1.1 allows the combination of MPL-licensed code with code licensed under different licenses, which is a distinction from the GPL. However, it does not grant patent or trademark licenses beyond the scope of the MPL-licensed code.

Section 3.5 explains the requirements for including notices in the Covered Code. According to the MPL 1.1, you must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to include the notice in a particular file due to its structure, you should include it in a relevant location (such as a directory) where users would be likely to look for such a notice.

If you make modifications to the Covered Code, you have the option to add your name as a Contributor to the notice described in Exhibit A. Additionally, you must duplicate the MPL 1.1 license in any documentation for the Source Code where you describe recipients' rights or ownership rights related to the Covered Code.

Section 3.6 governs the distribution of the executable versions of the Covered Code. It requires compliance with Sections 3.1 through 3.5 and includes a notice stating that the Source Code version of the Covered Code is available under the terms of the MPL 1.1. The second part of Section 3.6 allows the distribution of the executable version under a different license than the source code, including proprietary licenses, as long as the license does not limit or alter the recipient's rights in the Source Code version as stated in the MPL 1.1. If the Executable version is distributed under a different license, it must be made clear that any differing terms are offered solely by the distributor and not by the Initial Developer or any Contributor.

Section 3.7 addresses the creation of Larger Works by combining the Covered Code with other code not governed by the MPL 1.1. It permits the distribution of the Larger Work as a single product, as long as the requirements of the MPL 1.1 are fulfilled for the Covered Code within the Larger Work.

Section 4 of the MPL 1.1 discusses the situation when compliance with the license terms becomes impossible due to statute, judicial order, or regulation. In such cases, the licensee must comply with the terms to the maximum extent possible and describe the limitations and affected code in the LEGAL file described in Section 3.4. This description must be included with all distributions of the Source Code.

The remaining sections of the license address topics such as license versions, derivative works, termination, disclaimer of warranty, limitation of liability, U.S. government end users, and miscellaneous provisions.

Please note that the information provided is a summary of the MPL 1.1 based on the section you shared. For a comprehensive understanding of the license, it is important to review the complete text and consult with legal professionals when necessary.

3.4

The GPL license, along with its offshoots like the LGPL, has had a significant impact on the development of software, particularly exemplified by the GNU/Linux project. Linux, which is built on the GPL license, has become a major competitor to the Windows operating system. The success of the GPL can be attributed to its symbolic and practical effects.

Symbolically, the GPL embodies the philosophy of free software. It ensures that contributions to GPL-licensed projects remain "free" in the sense of being available to other programmers and the community. Programmers who align with the ideals of open software development are drawn to the GPL license, resulting in a thriving community of contributors. The GPL's aspirational purpose attracts those who take open development seriously, shaping the nature of software development.

Practically, the GPL's restrictions have influenced the development of GPL-licensed software. Programmers generally respect the GPL and adhere to its terms, which has contributed to the success of GPL-licensed projects. The GPL's requirement that contributions be made available to the community fosters collaboration and the sharing of knowledge.

It's important to note that the GPL does not require software running on a GPL-licensed operating system to be licensed under the GPL. Similarly, the GPL does not mandate that only GPL-licensed programs be distributed alongside GPL-licensed code. However, some programmers and distributors disfavor such distributions, believing they are inconsistent with the spirit of free software development.

An example of the GPL's impact on software development is the development of KDE and GNOME, both graphical user interface frontends for GNU/Linux. KDE was initially based on the non-GPL licensed Qt Toolkit, which sparked controversy among those who favored free software. As a result, a separate team developed GNOME, which adhered to the GPL. Both KDE and GNOME have thrived and become widely supported.

Overall, while the GPL sets the foundation for free software development, it also represents a larger idea of how software should be created and maintained. Its success stems not only from its legal terms but also from the ideals it promotes and the community it attracts.

The Mozilla Public License (MPL) was introduced when Netscape Communications decided to open source license its Netscape Communicator software in 1998. At that time, Netscape was in fierce competition with Microsoft's Internet Explorer, which had the advantage of being closely tied to the dominant Windows operating system. The MPL was an attempt to leverage the benefits of open source development while still maintaining a program developed under a proprietary license.

The initial announcement of opening the Communicator code generated significant enthusiasm and helped enhance the public perception of open source software. However, Netscape's financial situation and its eventual acquisition by America Online posed challenges for the project. Nevertheless, through extensive rewriting and a focus on standards compliance, a vibrant Mozilla culture emerged, somewhat independent of its Netscape roots. The open sourcing of the project did not reverse Netscape's fortunes, but it became a crucial source of ongoing innovation in the web browser market. This was further demonstrated with the release of the Firefox web browser by the Mozilla Foundation in 2004, showcasing the continued vitality of the project.

The MPL itself has thrived as an open source license. It is well-constructed and well-written, and it has found its niche in the open source community. Only the BSD, GPL, and LGPL licenses are associated with a larger number of projects than the MPL. Additionally, the MPL has served as the foundation for several other Open

Source Initiative-certified licenses, such as the Apple Public License, Nokia Open Source License, and Sun Public License.

The success of licenses like the GPL and the MPL is not solely dependent on the specific terms or wording of those licenses. Rather, it is the ideas they represent that attract attention and drive success. Powerful and meaningful ideas have the ability to attract minds and inspire innovation. The achievements of open source and free software licensing are a testament to the impact of such ideas and the communities they foster.

Qt, Artistic, and Creative Commons Licenses

4.1

The Q Public License (QPL) was created by Trolltech, a Norwegian company, to govern the distribution of its software, the Qt Toolkit. The Qt Toolkit is a cross-platform toolkit used for developing graphical user interface (GUI) applications and is widely used in the KDE desktop environment for UNIX and UNIX-like operating systems, including Linux.

As KDE gained popularity, concerns arose within the open source and free software community regarding the limitations imposed by the QPL. In response to pressure from the community, Trolltech decided to cross-license the Qt Toolkit under the GPL in addition to the QPL. As a result, the developers of KDE immediately transitioned their license to GPL. Since then, the distribution of the Qt Toolkit and KDE has predominantly been under the GPL.

The QPL introduced a novel approach to various open source licensing issues. It allows the distribution of modifications to the licensed software in the form of patches under less restrictive terms compared to modifications compiled with the original code. It also provides certain rights that apply only to the initial developer of the licensed code.

The QPL is structured in numbered sections following an introduction. The introduction establishes the copyright notice, permits the distribution and copying of the license, and states that the license applies to all software containing the appropriate copyright notice. It also clarifies that any software based on or using the software must comply with the terms of the QPL.

Sections 1 to 6, titled "Grant of Rights," outline the rights granted under the QPL. Section 1 emphasizes that acceptance of the license is indicated by the distribution of the software and specifies that any work incorporating, relying on, or linking to the software must adhere to the terms of the license. Section 2 grants the right to distribute unmodified versions of the software, provided that all relevant notices and disclaimers are included. Section 3 permits the distribution of modifications as separate patches.

The QPL's distinction between distributing modifications as patches and distributing modified executable code provides benefits to the original developer, users of the modified software, and contributors. By distributing modifications as patches, it clearly distinguishes the work of the initial developer from that of contributors, protecting the reputation of the former and emphasizing their primary contributions.

From a user's perspective, the benefit of distributing modifications as patches is that users have access to the functionality embedded in the original work without having to compile the patches themselves. However, if users want to access functions or improvements made by contributors, they would need to recompile the source code to include the patches, which can be a manageable task.

Another example of a product that allows free distribution but prohibits changes is the qmail mail server. Developers extending qmail also use patching methods, although there aren't explicit rules on the nature of those patches.

However, from the contributors' point of view, distributing modifications as patches presents a drawback. Since users are less likely to use the contributor's version of the software than the initial developer's version due to the additional effort required, contributors may choose to distribute software as a patch because the QPL requires them to surrender fewer rights to their work compared to distributing modifications as executable code.

Section 3 of the QPL places limited restrictions on modifications. It states that modifications must not alter or remove any copyright notices in the software, and it grants the initial developer a non-exclusive royalty-free right to distribute the modifications in future versions of the software. Contributors are not required to surrender any other rights and can license the patch under any license, as long as compliance with Section 3(a) and (b) is maintained. The source code for the patch is not required to be made available.

Section 4 governs the distribution of executable code, both with and without modifications. It requires the inclusion of the license document in the distribution, ensures that recipients can receive the complete source code without any additional charge, and mandates that modifications included in the distributed executable code be available under the terms of the QPL.

Sections 5 and 6 address the combination of the software with other products and the distribution of linked software. Section 5 permits users to compile, link, and run application programs developed using the software, while Section 6 sets requirements for the distribution of application programs, reusable components, and other software items linked with the software. These items must be accompanied by the source code, licensed for use and redistribution without charges, and made available to the initial developer upon request if the distribution is non-public.

The QPL includes provisions for limitation of liability and disclaims warranties, similar to other open source licenses. It also contains a choice of law and choice of forum provision, which may need revision if adopted outside of Norway.

With the cross-licensing of the Qt Toolkit under the GPL and the adoption of the GPL by KDE developers, the importance of the QPL as a license may diminish.

4.2

The Artistic License, also known as the Perl Artistic License, is a software license that was developed by Larry Wall in the late 1980s for the Perl programming language. It is designed to allow the initial developer to maintain control over the software while granting users the right to use, modify, and distribute the software.

The license begins with a preamble that states its intention to give the Copyright Holder artistic control over the software while still allowing users to use and distribute it. It also includes definitions of terms used in the license, such as "Package," "Standard Version," "Copyright Holder," and "You."

The license provides several options for copying, modifying, and distributing the software. Section 1 allows the verbatim copying of the source code of the Standard Version of the Package. Section 2 permits the application of bug fixes, portability fixes, and other modifications derived from the public domain or from the Copyright Holder, while still considering the modified package as the Standard Version.

Section 3 addresses modifications that are not part of the Standard Version and requires them to be clearly marked. The modifier must choose one of several options, including placing the modifications in the public domain or making them freely available, using the modified package only within their organization, or renaming non-standard executables and documenting the differences.

Section 4 governs the distribution of the software in object code or executable form. Distributors must choose one of several options, such as distributing the Standard Version of the executables and library files, providing the machine-readable source code with modifications, or providing non-standard executables with their corresponding Standard Version executables.

Section 5 allows distributors to charge a reasonable copying fee for the distribution of the software but prohibits charging a fee for the software itself. However, it permits the distribution of the software as part of a larger software distribution, as long as the distributor does not claim it as their own product.

Section 6 clarifies that scripts and library files generated by the software do not fall under the copyright of the package but belong to whoever generated them. Section 7 excludes C or Perl subroutines supplied by the user from being considered part of the package.

Sections 8 and 9 contain non-endorsement clauses and disclaimers of warranty, respectively. The optional Section 8 in the Perl Artistic License permits the aggregation of the package with a commercial distribution, as long as the use of the package is embedded and not made visible to the end user.

While the Artistic License is specifically designed for Perl, it can be used apart from Perl as well. The license is known for being vague and confusing in certain aspects, but it is commonly encountered due to Perl's popularity and its association with the language.

4.3

the Creative Commons licenses were created by the Creative Commons Corporation, a nonprofit organization founded in 2001 and based at Stanford University Law School. The licenses were designed to encourage creators to share their works with the public while providing a solid basis for licensing various types of creative expressions beyond just software.

The Creative Commons licenses allow free copying and distribution of the licensed works, and some variations also permit the creation and distribution of derivative works. The Attribution-ShareAlike license, described in the passage, is one of the licenses offered by Creative Commons. It allows the distribution of the original work and derivative works under the same license terms. It also requires attributions to the original author of the work.

Additionally, the Creative Commons Corporation provides other services such as the Public Domain Dedication, which allows creators to dedicate their works to the public domain, and the Founder's Copyright, which mimics the effect of the original copyright laws by granting copyright for a limited period.

The passage also mentions that the Creative Commons licenses are well-constructed and provide a good model for those considering drafting their own open source licenses. It notes that the licenses are not written specifically for software but can be applied to various forms of creative expression.

Section 5(b) includes a disclaimer of warranties, stating that the work is licensed "as is" without any warranties regarding its contents or accuracy. Section 6 limits the liability of the licensor, except for damages resulting from breaches of warranties. Section 7 covers the termination of the license, specifying that a breach by the

licensee leads to termination but does not affect the license for those who received derivative works from the licensee and remain in compliance. Section 8 contains miscellaneous provisions such as reformation of invalid provisions, written waivers or consents, and the agreement's entirety.

The text also mentions the addition of new features in the Attribution-ShareAlike Version 2.0 license, including requirements for attribution to the original author and the ability to "relicense" the work under different terms or stop distributing it. It discusses restrictions on altering the terms of the license, the inclusion of copyright notices and disclaimers, and the need to credit the original author in derivative works. The text further touches on fair use rights, license grants, restrictions, representations, warranties, limitations on liability, termination conditions, and miscellaneous provisions.

Overall, the Creative Commons licenses provide a legal framework for sharing and using creative works under certain conditions, while granting copyright holders some control and protection over their works.

Non-Open Source Licenses

5.1

The classic proprietary license described in the provided text is a standard license commonly used for proprietary software. It is a "shrinkwrap" license typically seen in single license sales to individual users. Here's a breakdown of the key points and provisions of the license:

General: The license states that the software is licensed, not sold, to the user. The user owns the media on which the software is recorded, but the ownership of the software itself remains with the licensor (Mildew Corporation).

Permitted License Uses and Restrictions: The license allows the user to install and use one copy of the software on a single device at a time. The user is prohibited from installing the software on multiple devices or making it available over a network.

Transfer: The user is not allowed to rent, lease, lend, sublicense, or distribute the software. However, the user can make a one-time permanent transfer of their license rights to another party, provided that the recipient agrees to the terms of the license.

Termination: The license is effective until terminated. If the user fails to comply with the terms of the license, Mildew can terminate the license without notice. Upon termination, the user must cease all use of the software and destroy all copies.

Limited Warranty on Media: Mildew warrants that the media on which the software is recorded is free from defects for a period of 90 days from the date of purchase. The user's exclusive remedy is a refund or replacement of the defective media.

Disclaimer of Warranties: The software is provided "as is" without any warranties, either express or implied. Mildew disclaims all warranties regarding the software's quality, performance, accuracy, and fitness for a particular purpose.

Limitation of Liability: Mildew is not liable for any personal injury or any incidental, special, indirect, or consequential damages arising from the user's use or inability to use the software. Mildew's total liability is limited to fifty dollars ($50), except where prohibited by law.

Export Law Assurances: The user may not export the software to certain embargoed countries or individuals listed by the U.S. Treasury Department or Department of Commerce. The user represents and warrants that they are not located in such countries or on such lists.

Government End Users: The license terms also apply to U.S. government end users. The software is considered a commercial item, and the government end users have the same rights as other users.

Controlling Law and Severability and Choice of Forum: The license is governed by the laws of the State of Colorado, and any disputes must be brought before courts in Colorado. If any provision of the license is deemed unenforceable, the remaining provisions remain in effect.

Third Party Notices and Conditions: The software may include software owned by Mongrel Mix, which is licensed under the MIT License. The terms of the MIT License apply to those portions of the software derived from Mongrel Mix.

Complete Agreement; Governing Language: The license constitutes the entire agreement between the parties and supersedes any prior understandings. Any amendments or modifications must be in writing and signed by Mildew.

5.2

The SCSL was developed by Sun Microsystems (now Oracle) for Jini and Java to incorporate some open source principles while maintaining certain restrictions.

The SCSL is not characterized as an open source license and has significant differences from open source licenses.

The SCSL includes a compatibility requirement, meaning modified versions of the licensed work cannot be deployed without certification by the licensor (Sun).

Commercial use of the SCSL-licensed code may require the payment of royalties, which is inconsistent with the open source model.

The SCSL introduces the concept of "Community Members" who agree to be bound by the license.

Definitions are provided for terms such as Community Code, Contributed Code, Error Corrections, Interfaces, Modifications, Reference Code, and more.

The license grants rights to Community Members for research use, including reproducing, modifying, and distributing the Reference Code and Technology Specifications.

Contributed Code can be shared with other Community Members, and the license grants rights to the Original Contributor to use, reproduce, modify, and distribute Contributed Code.

There are obligations for licensees, such as providing source code, reproducing notices, documenting modifications, and adhering to the license when using and distributing Modifications.

Regarding distribution, object code can be shared with third parties for research purposes only, as long as the license chosen is consistent with the SCSL. Extensions allow Community Members to create interfaces, but they must not incorporate Reference Code without permission. If interfaces are disclosed for independent development, specifications need to be published and made available for testing.

The license includes provisions related to intellectual property rights, stating that developers of interfaces should not assert IP rights that would be infringed by independent implementations, except for their specific implementation.

Termination of the license can occur at the discretion of either the Community Member or the Original Contributor. Non-compliance with material terms may lead to termination. Patent litigation against a Community Member may result in the termination of specific patent licenses, while litigation against the Original Contributor can lead to the termination of all rights.

The SCSL contains limitations of liability, disclaiming responsibility for third-party claims based on IP rights infringement. Liability for breach or tort is limited to a specified amount, and no liability is assumed for indirect or consequential damages.

The miscellaneous provisions cover topics such as trademark usage, integration and assignment of the license, severability of provisions, governing law, dispute resolution through arbitration, U.S. government acquisition, and international use. The SCSL prohibits commercial use of the licensed code, allowing only for research, evaluation, development, educational, or personal and individual use. Any use or distribution for direct or indirect commercial gain or advantage is not permitted.

However, the SCSL does provide a separate Commercial Use Supplement that allows for commercial use of a different category of code. This supplement is an additional license and covers code that has passed through the research and development phase and has been deemed compliant with the code standards.

Under the Commercial Use Supplement, certain rights are granted for commercial use, subject to compliance with the terms and conditions of the research use license and the supplement itself. These rights include using compliance materials to determine if the code is a compliant implementation, reproducing and distributing compliant implementations internally and to third parties, distributing source code to community members for commercial use, and distributing the technology specifications with compliant implementations.

The Commercial Use Supplement imposes additional restrictions and community responsibilities. For example, certifications of status may be required when distributing source code, and compliance with the specified compliance materials and technology-specific requirements is necessary. The licensee must ensure that only compliant implementations are used and distributed for commercial use.

The Commercial Use Supplement also addresses licensing terms, distribution requirements, defense of claims, notices of breach or infringement, proprietary rights notices, and assignment restrictions.

Overall, the SCSL and its Commercial Use Supplement combine open source and proprietary contract ideas and values. While the SCSL restricts commercial use, the supplement allows for such use with certain limitations and requirements, including the potential payment of royalties and compliance with Sun's specifications. The SCSL aims to integrate some aspects of open source into a commercial model while emphasizing operational uniformity.

5.3

While Microsoft has historically relied on proprietary software licensing, it has faced pressure to provide access to its source code due to the rise of open source and free software.

Microsoft's response to this demand is the Microsoft Shared Source Initiative, which encompasses various software licensing practices. It aims to facilitate access to Microsoft source code while also serving as a

lobbying effort to defend strong intellectual property laws.

The Shared Source Initiative offers different source code licensing attributes, allowing varying levels of access and modification rights to the source code. The licenses are tailored based on factors such as the licensee's country, the importance of the product to Microsoft's core business, and the purpose of use (commercial, charitable, academic).

However, the Shared Source Initiative is complex and lacks the simplicity and transparency of open source licenses. It is considered an extension of Microsoft's commercial licensing practices rather than a true open source initiative. While it has been beneficial within Microsoft's ecosystem, external reactions have been more ambivalent, as the licenses often impose restrictive terms on core application or operating system code.

Nevertheless, Microsoft has been observing developments in the open source movement and has started using open source licenses for some of its newer projects. This suggests a recognition of the advantages that open source offers, even for Microsoft. The company is also exploring ways to integrate open source approaches with its traditional business practices, which may result in changes to existing open source practices through litigation, lobbying, and cooperation.

Although Microsoft positions its shared source approach as an alternative to open source, it is important to monitor its actions in this area. Microsoft's release of WiX technology under an Open Source Initiative-approved license indicates a potential shift, but the extent of its future involvement in open source remains to be seen.

Overall, Microsoft's approach falls on the proprietary end of the licensing spectrum, given its historical reliance on proprietary software licensing. However, the Shared Source Initiative shows some movement towards open source models, making it a noteworthy project to watch.

Legal Impacts of Open Source and Free Software Licensing

6.1

In contract law, consideration refers to the mutual obligations created by an agreement. For a contract to be enforceable, each party must undertake some obligation as part of the transaction. Without consideration, a promise is not legally enforceable. However, even the most unrestrictive open source license, such as the MIT License, imposes a minimal obligation on licensees, such as including copyright and permission notices. This obligation constitutes consideration, making the license an enforceable contract.

Mutual consent, or a meeting of the minds, is another essential element of a contract. In ordinary commercial contracts, mutual consent is usually straightforward, as parties negotiate and reach a final agreement documented in a signed contract. However, open source and free software licenses present a more complex problem due to the absence of a formal signed document.

Different types of licenses, such as "browsewrap" and "clickwrap," determine the level of mutual consent required to create an enforceable contract. In a browsewrap license, the user is informed that the software is subject to a license but is not required to assent to the terms before accessing the software. The enforceability of browsewrap licenses is subject to dispute and may not always result in an enforceable contract.

On the other hand, a clickwrap license requires the user to view the license terms and take affirmative action, such as clicking a button, to indicate consent before accessing the software. Clickwrap licenses are more likely to create an enforceable contract as they provide sufficient notice of the terms and require explicit consent.

The excerpt also mentions a variant of clickwrap and browsewrap licenses where the user only views the license without taking affirmative action. While the user is aware of the license and its governing terms, the absence of affirmative consent raises concerns for courts. It is seen as unfair to enforce terms when one party has not positively affirmed their consent.

The issue of mutual consent is relevant to open source and free software licenses. In some cases, users may come across code or software subject to these licenses without having to take explicit action to indicate their consent. This raises questions about the enforceability of the license in such situations.

Overall, the principles of consideration and mutual consent play a crucial role in determining the enforceability of contracts, including open source and free software licenses. The specific form and requirements of the license, such as browsewrap, clickwrap, or implied consent, can impact the enforceability of the license as a contract.

6.2

The Uniform Electronic Transactions Act (UETA) and the federal law called E-Sign recognize the legal validity of contracts recorded in electronic form, stating that they should not be denied legal effect simply because they are not on paper. E-Sign also acknowledges the equivalence of digital signatures to traditional written signatures. These laws do not aim to change the interpretation of contracts under state law. On the other hand, the Uniform Computer Information Transaction Act (UCITA) modifies state contract laws concerning software transactions. However, UCITA has only been adopted by Maryland and Virginia, while other states have implemented anti-UCITA statutes, limiting its impact.

6.3

Open source and free software licenses, such as the GPL and MIT License, impose restrictions on the rights granted under the license rather than affirmative obligations on licensees. These licenses are self-enforcing because users who wish to exercise the rights granted by the license must adhere to its terms, while those who disclaim the license find themselves without the fundamental rights granted by the license. Enforcing these licenses is relatively easy, as infringers risk losing their rights to the licensed code, and the licensor can contact customers of illegally licensed software to inform them about the original license terms and encourage compliance. The Free Software Foundation has successfully enforced the GPL License using similar methods.

6.4

Enforcement of open source and free software licenses in jurisdictions outside the United States presents challenges due to the lax enforcement of international copyright laws. However, in many developed countries where software creation takes place, copyright enforcement is routine and copyright protection is more stringent than in the United States. The existence of a stable organization that provides reliable software and support is crucial, and such organizations are subject to legal enforcement of copyright law. The enforcement of open source and free software licenses outside the United States is possible, with the Berne Convention serving as a framework for copyright protection. While some unauthorized distribution or "pirating" may occur, it does not necessarily undermine the successful distribution of licensed work. The nature of "pirating"

in software is limited, as it primarily involves the distribution of unmodified versions without payment, which is often allowed by open source licenses. Distribution of modified work in violation of the license terms is considered a violation. The major software markets are subject to legal and practical constraints, limiting the extent of piracy compared to other forms of media. Overall, market forces and social dynamics help limit infringement of open source and free software licenses, even without vigorous legal enforcement by the licensor.

6.5

Violating open source and free software licenses can have severe consequences for the violator, particularly regarding their ability to enforce their own copyrights. When a violator fails to comply with the terms of a license, they jeopardize their own ability to enforce copyrights related to the modified work they have invested in. For example, the violation of the MIT License's requirement to include copyright and permission notices can undermine future copyright enforcement. If a violator infringes upon a license and later seeks to enforce their own copyright, they may encounter the equitable doctrine of unclean hands, which can defeat a copyright claim if it was obtained unfairly or inequitably. Violations of licenses also expose the violator to potential lawsuits from the creators of the original work, who may seek damages or invalidate the violator's copyright. Violations of "copyleft" licenses, such as the GPL, can have even more severe consequences as they require derivative works to be subject to the terms of the license. Failure to comply with the GPL voids any rights granted by the license, and continuing to distribute a program under a proprietary license after violating the GPL can potentially involve lawyers in the violator's wrongdoing. These consequences apply to any distribution of licensed software that is inconsistent with the terms of the open source or free software license being violated.

6.6

The open source and free software communities play a crucial role in the enforcement of open source and free software licenses. Beyond legal and practical reasons for compliance, there is a fundamental moral principle that drives most programmers to adhere to these licenses. The code developed under these licenses is primarily the work of volunteers who have dedicated their time and efforts to benefit others. These programmers have often sacrificed more lucrative opportunities in the commercial software industry. Behind the legal terms of the licenses lies the principle that free code is a social good in itself, and this principle is deeply felt by the community.

This strong moral principle serves as a powerful motivator for compliance with open source and free software licenses. Violating these licenses and taking someone else's work as one's own is considered unthinkable and a gross violation of the community's principles. Even those who have not internalized this principle have incentives to comply due to the potential consequences of violating the norms of the community. Violators may face ostracism, being ignored or criticized in online forums, and exclusion from projects involving community members.

While there are ideological differences within the open source and free software communities, there is considerable common ground on the principle that taking and distributing someone else's work without regard for the creator's intent is wrong. However, it should be noted that movement from open source projects to proprietary projects, as long as it is done in a way consistent with the original license and principles, does not generate ill feelings within the community.

In summary, while contracts and legal enforcement are important for protecting open source and free software licensing principles, the true guardians of these principles are the programmers and users themselves. The deeply held moral principle and the collective norms of the community contribute significantly to the compliance and enforcement of these licenses.

6.7

When a programmer wants to combine elements from two or more programs that are under different licenses, it is important to determine whether these licenses are compatible. It cannot be assumed that licenses are compatible simply based on their appearance or similarity. Even seemingly innocuous terms in one or both licenses can render them incompatible with each other. Distributing or modifying programs that contain incompatible code would result in copyright infringement.

Analyzing the compatibility of licenses requires careful examination of the specific language and terms of each license. It should be noted that the licenses discussed in the book often serve as templates, and the actual language and terms may vary. Therefore, it is necessary to read the licenses involved thoroughly.

There are certain scenarios where the compatibility is clear. If one of the works is licensed under a proprietary license, it cannot be combined with code from another license (except through cross-licensing, discussed later). Proprietary licenses typically grant limited rights to use the software and do not allow modification or distribution. GPL-licensed code is generally incompatible with most other licenses because the GPL prohibits imposing further restrictions on recipients' exercise of the granted rights.

Public domain works, which have no copyright restrictions, can be combined with code from any license as long as the terms of the other license are complied with. However, precise guidance on compatibility between licenses is challenging to provide. Many licenses described in the book are templates with variations in their terms. Therefore, caution is necessary when considering the combination of code governed by different licenses.

Fortunately, cross-licensing offers a solution to compatibility issues. Cross-licensing involves obtaining permission from the original licensor to use the code under a different license. The original licensor retains all rights associated with the copyright and can choose to license the work under different terms. Open source and free software communities generally favor cross-licensing to promote free distribution and avoid duplication of effort. Some licenses, like the GPL, explicitly allow for cross-licensing. Approaching authors or licensors with cross-licensing requests is generally well-received within the community.

However, cross-licensing may not be a practical option in all situations. Projects with a large number of contributors may pose logistical challenges for cross-licensing. In such cases, relicensing the contributions of numerous individuals can be complex and impractical. Nonetheless, for most open source and free software projects maintained by a small group or individual, cross-licensing can often be obtained by simply asking for permission.

In summary, compatibility between licenses should not be assumed based on appearances. Careful examination of license terms is necessary, and cross-licensing provides a reliable solution for combining code from different licenses.

Software Development Using Open Source and Free Software Licenses

7.1

Initially, the Free Software Foundation, founded by Richard Stallman, did not rely heavily on the open development model due to limited access to the Internet in the early 1980s. Stallman himself took on much of the work for the GNU project and closely supervised the development. However, this approach had its limitations, and the GNU project faced challenges in completing its kernel, known as Hurd.

In contrast, Linus Torvalds started the Linux project in 1991 and introduced a different development model that became known as the "Linux development model." This model relied on user involvement, frequent releases, and an open and collaborative development process. Users played a crucial role by providing bug reports and patches, which led to faster bug fixes and improvements. The development model encouraged a large number of users, debuggers, and programmers to contribute, with notable individuals like Alan Cox, Dave Miller, and Ted Ts'o taking on significant roles.

Eric Raymond, in his essay "The Cathedral and the Bazaar," highlighted the significance of the Linux development model. It emphasized the importance of user engagement, the "release early, release often" strategy, and the advantage of having a diverse user base for effective bug identification and resolution. This model allowed for an open and collaborative development process, avoiding the limitations of closed development models seen in commercial software development.

The availability of the Internet and its resources, such as software archives and communication channels like email and Usenet groups, played a vital role in enabling the Linux development model to succeed. The combination of the legal infrastructure provided by open source licenses, such as the GPL, and the technical infrastructure of the Internet facilitated the growth of open source and free software projects.

It is important to note that the development model employed in a project is influenced by various factors, including the project's circumstances, the individuals involved, and the available technology. While the choice of a specific license is important, it is not the sole determinant of a project's success. The open development model can keep code open even if the governing license permits closing it, as demonstrated by IBM's adoption of Apache and its continued openness despite having the option to use a proprietary license.

7.2

Forking occurs when a project splits into separate development paths, resulting in distinct versions of the software. While forking can have healthy consequences in some cases, it can also lead to undesirable outcomes and undermine the foundation of open development.

The passage provides an example of forking within the GNU Emacs project, where a group of developers led by Jamie Zawinski formed a separate line of development apart from Richard Stallman and the GNU project. Personality conflicts, concerns about the progress of Emacs development, and differing views on future development contributed to this fork. As of the time the passage was written, both Emacs development projects continue independently.

Forks in mature projects are generally feared because they divide the user base and the community of contributors. Over time, the diverging development paths make it increasingly difficult to port modifications and bug patches between the forks. This duplication of effort and division of the development community can be detrimental, especially when the programs are quite similar.

While no open source or free software license can completely prevent forking, different licenses provide varying levels of protection against forks. Some licenses, such as the Apache License and Perl License, rely on

the reputation of the project developers to avoid forks but include terms that reinforce this defense. Other licenses, like the GPL, hinder forking by requiring that developers distribute or modify the code under open development terms. However, licenses such as the MIT or BSD License, which permit non-open development, may be more prone to forking.

The passage also mentions that forking can be prevented or decreased by protecting the project's name. The Apache License, for instance, includes provisions that prevent the use of the Apache name without permission, helping to maintain the integrity and centralized development of the project.

Ultimately, the prevention of forking is more reliant on project development and community dynamics than the specific choice of license. The open development model itself, while fostering collaboration and innovation, also creates an environment where forking can occur.

7.3

The choice of an open source or free software license is often influenced by circumstances rather than solely relying on the discretion of a programmer. When getting involved in an existing open source project, such as submitting patches or bug reports, the consideration of the project's license is usually secondary. Users and contributors are more concerned with the benefits of their contributions and the improvement of the program.

In the case of maintaining or leading an open source project, the license applicable to the development is typically inherited from the project's original license. Switching to a different license can be administratively and legally challenging, requiring the consent of all previous contributors. Even for new projects, the choice of license may be influenced by similar projects that already exist. Prior work on similar projects provides a foundation, and the developer needs to carefully consider the license applicable to the pre-existing project.

The choice of license may also be constrained by the licenses chosen by predecessors. This is particularly evident with the GPL license, which aims to have a "viral" effect on licensing decisions. When starting from scratch or working with code whose license can be changed, the decision on the license becomes a matter of personal preference. Factors that can influence this decision include the popularity, comprehensibility, and philosophy of the license, especially regarding compatibility with other licenses, including proprietary ones.

Developers are encouraged to consider using well-known licenses within the development community to make the project more transparent and reduce barriers to entry for contributors. Clear and well-written licenses, such as the BSD, MIT, Apache, and MPL licenses, are recommended. It is advisable to avoid licenses that are ambiguous, confusing, or poorly written.

The most significant decision in choosing a license often revolves around the choice between the GPL license and a less restrictive license. The GPL promotes open development models and discourages reliance on proprietary software. It creates an incentive for developers to license their code under the GPL by granting access to all code already licensed under the GPL. On the other hand, less-restrictive licenses, like the MIT, BSD, Apache, and MPL licenses, support the view that open development models can coexist with other software development models, including proprietary ones.

In summary, there is no definitive answer to which license is best for a given project. The choice depends on various circumstances and the preferences of the project developer.

7.4

Drafting a new open source license is not typically the best approach for most open source projects. Creating a new license involves additional time, expense, and can discourage potential contributors from participating in the project. Potential contributors who are concerned about licensing implications would need to read and understand the new license, which can be a significant barrier to entry, especially with long or complex licenses.

If someone chooses to draft a new open source license, the first step should be to retain a competent and experienced attorney with knowledge and experience in drafting legal documents, particularly open source software licenses.

The next step would involve devising the basic mechanics of the license. Consideration should be given to the intended function of the license, with a focus on the generational limitations placed on distribution and modification of the licensed work by licensees. Different licenses, such as the MIT, BSD, GPL, or MPL licenses, impose varying restrictions and requirements. The Open Source Definition, as described in Chapter 1, sets specific requirements for a license to be certified as compliant by the Open Source Initiative.

In addition to complying with the Open Source Definition, licenses should be drafted with clarity and avoid ambiguity or unusual terms. Including a definitions section can help avoid unnecessary repetition and ensure consistent language throughout the license. Disclaimers of warranties and limitation of liabilities clauses are commonly included in open source licenses to protect the licensor and contributors from liability.

Choice of forum and choice of law clauses are relatively uncommon in open source licenses but can be advantageous in certain situations, particularly for "developer-centric" licenses. These clauses allow the licensor to choose a local forum and the application of local law in the event of any dispute. However, consulting with a lawyer is essential to ensure that the chosen law will interpret and enforce the license as intended.

Addressing the applicability of patents to the licensed work is another important consideration. Including a clause that grants users a royalty-free right to any patents held by the licensor, and possibly subsequent contributors depending on the license terms, can help prevent patent litigation.

In summary, drafting a new open source license is not recommended for most projects due to the associated challenges and potential barriers to entry. Utilizing existing well-known licenses that comply with the Open Source Definition is generally a more practical and effective approach.