Licensing Code: A Primer for Software Engineers

Anjan Goswami

October 23, 2023

1 Introduction

Licensing is a pivotal aspect in the realm of software development, significantly influencing how software can be utilized, altered, and disseminated. Understanding the nuances of various licenses is paramount for developers, companies, and end-users, as the type of license a piece of software carries determines its accessibility, modification possibilities, distribution, and other usage terms. This document offers a comprehensive overview of both open and closed source licenses. It delves into the characteristics of prevalent open-source licenses, including MIT, Apache, and GNU GPL, outlining their relative freedoms and restrictions. Furthermore, it explores the landscape of closed source or proprietary licenses, underscoring the rights retained by original creators and the limitations placed on end-users. An informed decision between open and closed source licensing is crucial, considering the potential legal and financial implications, and the impact on software's adaptability, reach, and future development.

2 Open Source Licenses

2.1 MIT License

Permissiveness Highly permissive. Allows for modification, distribution, private use, and sublicensing without many conditions.

Requirements Requires the retention of the original license and copyright notice in the redistributed code.

Patents No explicit patent grant, but the permission to sublicense can be interpreted as an implicit patent grant.

Liability Contains a liability and warranty disclaimer.

Usage Commonly chosen for its simplicity and permissiveness.

2.2 Apache License 2.0

Permissiveness Permissive. Allows for modification, distribution, private use, and sublicensing.

Requirements Requires the retention of the original license and copyright notice, stating of changes when modified, and providing a NOTICE file if one is provided with the software.

Patents Explicitly grants patent rights from contributors to users, covering both their contributions and any derivative works.

Liability Contains a liability and warranty disclaimer.

Usage Popular for its patent grant provisions and explicitness in handling contributions.

2.3 GNU GPL v2

Permissiveness Copyleft license. Derivative works must also be licensed under GPLv2.

Requirements Modified versions must be marked and the original and modified versions made available when distributed.

Patents Implicit patent grant inferred from the act of distribution under GPLv2.

Liability Contains a disclaimer of warranties.

Usage Commonly used, though some projects have migrated to GPLv3 for its clearer terms.

2.4 GNU GPL v3

Permissiveness Copyleft license with terms that address tivoization and other concerns.

Requirements Like GPLv2, but also mandates providing installation information to let end-users run modified versions.

Patents Contains explicit patent grants and provisions against patent litigation.

Liability Contains a disclaimer of warranties.

Usage Adopted by projects that needed the explicit terms about patents and tivoization.

3 Closed Source Licensing

Closed source, or proprietary software, is defined by its exclusive legal rights which are reserved by the original authors or developers. These rights often restrict the use, modification, and distribution of the software. Here are some standard clauses and rights that can be present in a closed source license:

3.1 Standard Clauses

Grant of License Typically grants the licensee the non-exclusive right to use one copy of the software, on one system, for any purpose.

Restrictions on Use The licensee may not distribute, sell, lease, or rent the software to others. Reverse engineering, decompilation, and disassembly are often also prohibited.

Ownership and Proprietary Rights The licensor retains all ownership rights to the software, and the software is considered proprietary.

Termination of License Stipulates the conditions under which the license can be terminated, often involving the violation of any license provision.

Disclaimer of Warranty Proprietary software is often provided "as is," with no warranty of any kind.

3.2 Rights in the USA

In the United States, software is primarily protected under copyright law, giving developers the exclusive right to reproduce, distribute, perform, and display the software. Patent law can also protect specific algorithms or processes within the software. The developer, thus, has a strong legal construct to protect and enforce their rights against unauthorized use, distribution, or copying of their software. The Digital Millennium Copyright Act (DMCA) further bolsters these rights by criminalizing the circumvention of software access controls.

3.3 Rights and Issues in the EU

In the European Union, software is protected both under copyright and sui generis database rights. The latter provides protection for the structure, sequence, and organization of software. However, unlike the USA, EU laws often encompass user rights or "fair dealing" rights, which may allow for actions such as private copying or reverse engineering for interoperability purposes. Software patents are a complex issue in the EU; while computer-implemented inventions can be patented, algorithms or business methods cannot.

3.4 Rights and Issues in Asia

Asian countries have diverse approaches to software licensing. For instance, in China, software is protected under copyright law, and there's an increasing push towards recognizing software patents. However, enforcement can be challenging due to widespread software piracy. In India, while software is copyrightable, it can only be patented if it results in a new invention either by itself or in conjunction with hardware. Enforcement, again, remains a challenge in many Asian jurisdictions due to a combination of legal, cultural, and economic factors.

4 Recommendations

When choosing a license, it's crucial to consider your goals for your software. If you wish for the maximum number of people and projects to use it without restriction, a permissive license like MIT might be the best choice. However, if you want to ensure that derivative works also remain open, then a copyleft license like the GPL is more appropriate. For projects where patenting is a concern, the Apache License 2.0 is preferable due to its explicit patent grant. Always consult with legal counsel when making decisions about software licensing.

5 Case Studies on Misuse of Open Source Licenses

egal contract that specifies the terms under which software can be used, modified, and distributed. A lack of clarity or diligence in adhering to these terms can lead to significant consequences, both in financial and reputational terms. Moreover, as software becomes increasingly integrated into every facet of business and society, the stakes of such oversights become higher. It's thus of paramount importance for businesses and developers to fully understand and comply with the licenses under which they operate. To further underline this, we present two real-world cases where misunderstandings or misapplications of open-source licensing led to legal disputes. These cases serve as cautionary tales, highlighting the real-world implications of licensing missteps.

5.1 Case Study 1: Versata vs. Ameriprise

Date: 2012
Companies Involved: Versata Software vs. Ameriprise Financial Services
Reference: United States District Court for the District of Minnesota, Civ. No. 13-1211 (SRN/JSM)

Details: Versata Software sued Ameriprise Financial Services for breach of software licensing agreements. Versata had provided proprietary software to Ameriprise, who then hired a third party to support it. Ameriprise used open-source components that were subject to the GNU General Public License (GPL) to support the proprietary software. Versata claimed that

by doing so, Ameriprise had breached their contract, effectively turning their proprietary software into open-source software, and hence, they sought damages. The court determined that Ameriprise was in breach of the licensing agreement.

Source: `https://www.courtlistener.com/docket/4151926/versata-software-inc-v-ameriprise-f`

5.2 Case Study 2: Cisco vs. Free Software Foundation

Date: 2008
Companies Involved: Cisco Systems vs. Free Software Foundation

Details: The Free Software Foundation (FSF) sued Cisco, alleging that the company had infringed on FSF copyrights for multiple products under the GNU GPL and GNU Lesser GPL. Cisco redistributed FSF programs in its Linksys products but failed to provide complete corresponding source code, as required by the GPL. The case was eventually settled in 2009 with Cisco appointing a Free Software Director to ensure GPL compliance.

Source: `https://www.fsf.org/news/2008-12-cisco-suit`

6 Conclusion: The Upcoming Challenges in Software Licensing

The landscape of software development is evolving at an unprecedented pace, particularly with the advent of Language Model-driven code generation tools like OpenAI's Codex, which powers GitHub's Copilot. These tools, designed to augment a programmer's capabilities, can generate vast amounts of code based on patterns found in open-source projects.

The introduction of such AI-driven coding assistants ushers in a new era, where software development scales up significantly. However, this presents novel challenges in licensing. When AI generates code snippets, it's not always clear-cut from where the inspiration was derived, which could lead to inadvertent licensing infringements. This ambiguity can be problematic, especially if the generated code resembles open-source code with restrictive licenses, such as the GPL.

Furthermore, as AI begins to play a more significant role in creating software, the line between original human-made content and machine-generated content becomes blurred. This could lead to intricate legal challenges about software ownership, rights, and appropriate licensing.

In the dawn of this new era, it's imperative for developers, legal experts, and organizations to collaborate, ensuring that the next generation of software is both innovative and compliant with the evolving world of licensing.