Software Development Using Open Source and Free Software Licenses

Presentation: Assoc.Prof.Dr. Vu Thanh Nguyen
Msc. Nguyễn Công Hoan

Content

- Models of Open Source and Free Software Development
- Forking
- Choosing an Open Source or Free Software License
- Drafting Open Source Licenses

Introduction

- The purpose of open source and free software licensing is to permit and encourage the involvement by licensees in improvement, modification, and distribution of the licensed work.
- This open development model of software development is the unique strength of the open source and free software movement. While the open source and free software licenses already discussed approach open software development differently, open development is the goal.

Introduction

- This chapter describes the basic principles of software development under open source and free software licenses, including the problems of forking, community development under the bazaar and the cathedral models, how open source and free software projects are initiated and maintained, and the effect that license choices can have on software development.

Models of Open Source and Free Software Development

- The open source and free software licensing is driven by the development model, or models, that it is intended to encourage.
- These licenses are intended to permit, and indeed, to encourage the contributions of others to the project.
- Nonetheless, one of the first open development projects relied, at least at the beginning, on a relatively small number of closely-knit developers.
  - This project was Richard Stallman’s plan to develop a complete operating system modeled after the Unix operating system but written entirely in free code

Models of Open Source and Free Software Development

- In its traditional form, commercial software development is based on the exploitation of the monopoly created by copyright for competitive advantage. It makes sense in that system to avoid any process that would undermine that advantage, such as, for example, the sharing of source code with thousands of potentially competing strangers.
- Programmers for commercial concerns do “work-for-hire”: the code they write does not belong to them but to their employers. They are routinely required to sign non-disclosure agreements, preventing them from disclosing to anyone else information that is proprietary (i.e., what their employer considers to be proprietary).
- Such programmers are also frequently asked to sign non-compete agreements, which prevent them from working for their employer’s competitors for a year or two (or more) after they leave that employer.

Models of Open Source and Free Software Development

- This emphasis on secrecy channeled commercial programmers into cathedral-style models of software development. While such companies are free to hire as many programmers as they may need, even the resources of a company such as Microsoft are limited.
- No user base (or almost no user base) would be willing to subject itself to the disclosure restrictions that are required to maintain the commercial advantage software companies want. What results is a relatively small group of programmers, as talented as the resources and attractiveness of the company can gather, building the software project essentially in secret and presenting it as a black box to the software-buying public.
- This model of software development is not limited to commercial development.

Models of Open Source and Free Software Development

- The GNU project, while certainly not anywhere near as “closed” as traditional commercial software development, relied heavily on the contributions of a relatively small number of people who were relatively tightly organized.
- The GNU project did not, at least in its early days, follow a “release early, release often” model.
- GNU Emacs has incorporated the suggestions of hundreds of participants over more than 15 years of development and stands as a highly respected model of free software development. In addition, the GPL built a foundation for the open development model.
- The availability of software archives accessible by the Internet, Usenet groups open to contributors, and most importantly, email to permit communication between project originators, contributors, and users, **were all necessary for the success of the Linux development model on the scale that Linux itself has achieved.**

Models of Open Source and Free Software Development

- The legal infrastructure of open source combined with the technical infrastructure of the Internet to make this new approach possible.
- The Linux development model is obviously not the only one for developing software. It depends on the commitment and knowledge of its user base to succeed.
- End user applications (such as video games) have been slow to develop under open source or free software development models.
- Nonetheless, the Linux development model is useful (and powerful in its applications) for much more than just Linux itself. The same Linux-style development has been used successfully for a large number of programs.

Models of Open Source and Free Software Development

- The open development model may even keep code "open" that the governing license would permit to be closed, by incorporating it into a proprietary license.
  - ✓ For example, the Apache License permits distribution of modified versions under proprietary licenses. In June of 1998, IBM announced that it would ship Apache as part of its WebSphere group of programs and provide continuing enterprise level support for it.
  - ✓ The original Apache license permitted IBM to license its modifications under a proprietary license and not to disclose their source code, and the IBM Public License did nothing to limit its ability to do so.
  - ✓ Regardless of the terms of either of the applicable licenses, IBM's or Apache's, to get the full benefits of open source development, IBM had to live by open development rules.

Forking

- By maintaining its own Apache development as an open development project, IBM avoided creating a fork.
- Forking occurs whenever a software project splits. While the two versions may remain entirely or partially compatible for some period of time, inevitably the unique (and now distinct) histories of each one’s development will push them apart.
- Forks can happen for many different reasons and may have entirely healthy consequences.
- Such forks can occur without rancour and without any real concern for duplicative or unnecessary programming; the two future developments are so starkly different that mutual compatibility is of no concern.
- Forks in more mature projects, however, are much more capable of producing undesirable results.

Forking

- Accordingly, while the choice of license certainly can have some effect on preventing forks, the nature of the open development model is conducive to forking.
- Permitting open access to source code and encouraging development by outsiders both allows for and creates incentives for the development of forks.
- Addressing forks is less a question of adopting the proper license, as any open source or free software license permits forking in some way, and is more a question of project development.

Choosing an Open Source or Free Software License

- Choosing an open source or free software license is more often the result of circumstances than the unfettered discretion of a particular programmer.
- A typical route to involvement in an open source or free software project comes from contributing to an already existing project.
- Users frequently make even more substantial contributions to open development projects without much more consideration.
- A programmer may undertake even more substantial responsibilities for an open development project by helping to maintain it or even taking a leadership role, without choosing the license applicable to the development.
- In the world of open source and free software, projects are frequently handed down, and the "successor" lead programmer takes over a project from the project's initiator.

Choosing an Open Source or Free Software License

- Even for “new” open source or free software projects, the choice of a license may substantially be determined by license choices made by others. After all, given the nature of the open development model, it is frequently unnecessary to create a new program from scratch.
- In any event, prior work on similar projects in many situations will provide a foundation for a new project. In such a case, the developer has to consider carefully the license applicable to the preexisting project.
- Depending on the license, the developer may or may not have the ability to choose a different license to apply to the new project.
- Accordingly, in many situations, a developer’s choice of license is constrained by choices made by his predecessors. In fact, this is the intended purpose of one of the most popular of the open source and free software licenses, the GPL.

Choosing an Open Source or Free Software License

- In those situations in which a developer is in fact starting from scratch or from code whose license is amenable to change, the decision as to license will probably be largely a matter of personal preference.
- In choosing any license to apply to a new project, developers should strongly consider relying on those licenses already well-known in the development community.
- For much the same reason, using licenses that are written more clearly and which do not contain ambiguous or unusual terms will also help a project succeed.
- The most important decision in choosing a license will be the choice between a GPL License and a less restrictive license.

Choosing an Open Source or Free Software License

- The GPL is premised on the belief that non-free software is to be avoided and that free software development projects should be set up to encourage open development models of software development and to discourage reliance on software not developed under an open development model, including all proprietary-licensed software.
- By requiring that any code developed from or based on GPL-licensed code be GPL-licensed, the GPL creates a strong incentive for programmers to license their code under a GPL License, in the form of access to all the code already GPL-licensed.
- The argument in favor of less-restrictive licenses—such as the MIT, BSD, Apache, and MPL Licenses—is that open development model of software development is not inconsistent with the development of software under other models, including proprietary models.

Choosing an Open Source or Free Software License

- The more developers and users that are involved in working on particular code, the better, even if some of that development takes place in "closed" development models under proprietary licenses.
- In sum, there is no ready answer as to which license is the best for a given project. While a certain license may be better suited for a project, particularly when a substantial amount of work has already been done under that license, such decisions depend largely on circumstances and on the taste of the project developer.

Drafting Open Source Licenses

- Drafting a new open source license is probably not the best place to start for most open source projects. In addition to the extra time and expense associated with drafting any legal document, the use of a new license will discourage potential contributors from participating in the project.
- If you choose to do it, however, the first step in drafting an open source license should be retaining a competent and experienced attorney to undertake the task. While many open source licenses have been drafted by non-lawyers, the drafting of any contract, particularly one with the complexities inherent in open source software licenses, should be undertaken by someone with professional knowledge and experience.

Drafting Open Source Licenses

- After securing counsel, the next step should probably be devising the basic mechanics of the license. The new author should give serious thought to what the function of the license is intended to be. With open source and free software licenses, the key issue will generally be the generational limitations placed on distribution and modification of the licensed work by licensees.
- An open source license must permit an open development model to be applied to the licensed work, in that the source code must be provided or otherwise made available with the executable version of the code. The license must permit free modification of the licensed work and free distribution of both the original and the modified work. The license cannot discriminate in its application against any person or group of persons or any field of endeavor.

Drafting Open Source Licenses

- Disclaimer of warranties and limitation of liabilities clauses are virtually universal in open source licenses. Such clauses are not unique to open source licenses— many commercial software licenses contain similar terms.
- The use of choice of forum and choice of law clauses is relatively uncommon in open source licenses, but there are many situations in which such clauses could be advantageous to the licensor, particularly for "developer-centric" licenses, such as the **Apache License, v2.0, and the Perl License**.
- One final area that a developer should give some thought to addressing is the applicability of patents to the licensed work. In order to prevent patent litigation to the extent possible, it is probably worthwhile to include a clause in the license that grants specific permission for users to exercise a royalty-free right to any patents held by the licensor, and, depending on the terms of the license, any subsequent contributors.