Open Source Software Licensing

- Bruce Perens, bruce@perens.com
- Perens LLC

- One of the founders of the Open Source movement in software.
- Strategic consultant on Open Source issues to corporations, law firms, and governments.

- 510-904-3064

Open Source Licensing

- Provides critical rights: the right to *use*, *re-distribute*, *modify.*
- Open Source licenses *must* comply with the definition at http://opensource.org/osd.html or the licensed software is something less than Open Source.

Can You Open Source an Ontology?

- Copyright law probably can not be used to restrict the *use* of an ontology or the creation of an ontology that refers to another ontology.
- You can use Open Source licensing to *unambiguously license rights to others.*
- You probably can't use Open Source licensing to restrict “derivative works” of ontologies, because they're *not* derivative.
- Thus, the GPL and other reciprocal licenses probably can't be enforced on ontologies as they can on software.

What Can You Do?

- Enforce “moral rights” such as attribution and author-integrity (modified works must make clear that they aren't the author's work).
- Beyond that, you can try to get a patent, but this presents problems of prior art, etc.

What About Software?

- There is a much stronger legal foundation for the application of Open Source licensing to software.
- One U.S. enforcement case went to the Federal Circuit Appeals Court and was *not* sealed, and thus is of significant precedential value.
- All enforcement cases so far have dealt with a *total* failure of due diligence, not the more fine issues of derivative works.

Three Fundamental Kinds of Open Source Licensing

- *Gift* licensing: BSD, MIT, Apache.
- *Share-and-share-alike* or *reciprocal* licensing: GPL.
- Something in-between: LGPL

When to use Gift Licensing

- When you are creating a *standard* and *want everyone to do things your way,* whether it's in *proprietary* software (which grants less rights than Open Source) or Open Source.
- When you've already been paid to produce the work, and thus can afford to release all rights.
- When continuing control of the work isn't important.

When to use Share-and-Share-Alike Licensing

- When you want derivative works to themselves be Open Source.
- But be warned: there are ways to *circumvent* the restrictions of the GPL and create proprietary works that are not considered to be “derivative” in the law, but work as if they were. GPL is only imperfectly enforcible.

When to use Something In-Between *Gift* and *Share-and-Share-Alike*

- Software libraries which you wish to make available for use in proprietary products, but you want modifications to the library to be contributed back.
- Language interpreters, where you want to make it *especially* clear that they don't encumber the interpreted language or software into which they are embedded.
- Where continuing control of the original work, but not derivative works, is important.

Choosing a Suite of Licenses

- Choose three licenses for the three basic types: *Gift, Share-and-Share-Alike,* and something in between.
- Make sure the three are compatible with each other! Don't create legal complications in mixing your own software.
- You will still need to use other licenses when you participate in outside projects, as the project generally chooses the license.

Licenses of Note

- GPL – the prototypical share-and-share-alike license.
- GPL3 – a rewrite of earlier GPL versions to meet changes in the law since the 1980's. Takes on DRM issues, although I can show you how to use it legally without breaking your DRM. Provides *more* rights to combine with proprietary software than GPL2 does in some cases.

Licenses of Note

- Affero GPL – Handles the issue of derivative works that are *performed* but not distributed, as with Google. Comes in both GPL2 and GPL3 versions.
- Creative Commons – a set of licenses, some of which are Open Source and *some not.* The only right in common between the various Creative Commons licenses is the right to read the document.

Compliance

- When you modify or re-distribute Open Source, you must make sure to comply with the license terms.
- Reproduce Attribution.
- Convey the license and notices to others.
- Provide source code if the license requires that.
- License derivative works as required by the license.

Can You Stay in Control?

- Only while you are percieved as an effective developer and leader of the project.
- The rules explicitly permit forks of the project out of your control.
- Historicaly, forks have proven to be beneficial.

Dual Licensing

- Most famously used by MySQL.
- The copyright holder releases both a commercially-licensed version, and an Open Source version of the program under a maximally-restrictive license like Affero GPL 3.
- Commercial licensing is a “get out of jail” card from GPL terms.
- Has the complication that the copyright holder can't accept contributions without copyright assignment or the right to re-license.

Dual Licensing

- Developers historically have been reluctant to hand over their copyrights.
- But then, nobody's tried to give them any motivation to do so, other than the main developer's acceptance of the code.

Fallacies About Open Source

- Fallacy: Open Source restricts commercial use.
- Fact: Open Source licenses explicitly do not restrict *use,* which is running the program. They often do restrict the creation of proprietary derivative works.
- Fact: Open Source, because you can get it for free, can poison the potential to make much money from selling an identical program. But thery don't prevent income from service, etc.

Fallacies About Open Source

- Fallacy: You can't put Open Source software in a proprietary product.
- Fact: You can. There is *always* a way to separate your business-differentiating software from the Open Source so that it is not a derivative work. Get expert help.
- Fact: All of the cases of Open Source license enforcement so far have been regarding *stupid* non-compliance issues, like failure to distribute source code for the Open Source.

Necessary Policies

- Determine the acceptable Open Source licenses for the project. Make sure they're all compatible with each other.
- Create, and enforce, a patent policy for members that submit material (not just software, but standards committee input, etc.). Must at a minimum prevent submarine patents owned by project participants from encumbering the project.
- Look at W3C's policy.

Legal Homework

- Get a reality check from counsel on application of copyright law to ontologies.
- Wait, years or decades, for cases to make these issues more clear.

Questions

- Bruce Perens
- bruce@perens.com
- 510-904-3064
- Strategic consulting on topics of Open Source to corporations, law firms, and governments.