Open Source Licensing – Experience and Insights

Andrew Katz

Moorcrofts LLP

What do I do?

Advise on licence interpretation and compatibility

Help projects choose licences

Carry out licence audits

Draft open source policies/procedures

Advise on violations

Advise foundations and projects on governance and structure

Advise on licence migration

Try to avoid drafting licences

Introduction

In the beginning…

You paid for the hardware, not the software.

Photo courtesy IBM

In the beginning…

…computer users in academia, the military and commerce shared code, with no thought that it was somehow “protected”.

Then…this man got really annoyed..

…that after walking to retrieve his printout from a printer on a different floor, he found it was jammed. And he wasn't allowed access to the code to fix it. That wasn't sharing. It wasn't nice.

What's source code?

- Source code

```
//============================================================================
//            the list of all the possible states for the current FSM
//============================================================================
enum STATE{ START, INT, FLOAT, SCIENTIFIC, EXPONENT, S1, S2, INVALID } state;

STATE Transition( char *str );
void PrintState( STATE state );

int main() {
  // declaring buffer variable
  char buffer[32] = {0};
  // getting input from the user
  cout << "\nPlease enter a number: ";
  cin.getline( buffer, 32 );
  // compute final state
  STATE FINAL_STATE = Transition(buffer);
  // prints the final state
  PrintState(FINAL_STATE);
  return 0;
}
```

- Object code
  - 00101001011101010100101001000101010111

Software Freedom

- Richard Stallman (RMS)
- Free as in speech, not as in beer
- Freedom to use, study, share and adapt

Meanwhile…

This guy was working in his bedroom, starting a project to write an operating system kernel to complement to the rest of the work RMS was doing.

He called it “Linux”

The Linux Kernel

- Complements the GNU toolset, GCC etc
- GNU/Linux
- GPL2.0 – copyleft (from version 0.12)
- Classic “open source development model”
- But really uncharacteristic

Free Software vs Open Source

-Free Software Foundation vs Open Source Initiative

- Stallman vs. Perens/Raymond
- Authoritarian vs. liberal
- Moral stance vs. commercial
- GPL vs. BSD/Apache
- Pessimistic vs. optimistic

Cathedral vs. Bazaar

- Eric S. Raymond
- Cathedral = top down development
- Bazaar = self-organising system - emergence
- Non-hierarchical
- Hyper-meritocratic
- Release early, release often
- Many eyeballs make all bugs shallow

The Economic Model

How can anything free be any good?

Services, not licensing?

Do not open-source your key differentiator

A method of performing collaborative R&D

Minimises lock-in

(Open source and open standards)

Modularity

More free-market oriented than proprietary s/w

Some Stats from Carlo Daffara

- 44% of all code created in the world is OSS and increasing
- 80% of newly deployed code is open source
- 31% of OSX is OSS, 75% of Android.
- Stats demonstrate OSS more innovative than proprietary
- Open source shows slower growth of maintenance effort
- 36% lower defects in OSS than comparable proprietary code
- 80% of software is non-differentiating
- Revenue/employee in FLOSS firms: 221% of non-FLOSS

http://transfersummit.com/sites/default/files/materials/rgardler/ts11daffara-notes.pdf

http://www.openforumacademy.org/library/ofa-fellows-reference-library/ofe-fellows-reference-library/Hosted%20Files/first-conference-proceedingsA4.pdf

What is Free/Open Source?

- A philosophy
- A methodology
- A set of licences
- A business model
- None of the above/all of the above

Recap: software development

- Source code

```
//================================================================================
//            the list of all the possible states for the current FSM
//================================================================================
enum STATE{ START, INT, FLOAT, SCIENTIFIC, EXPONENT, S1, S2, INVALID } state;

STATE Transition( char *str );
void PrintState( STATE state );

int main() {
  // declaring buffer variable
  char buffer[32] = {0};
  // getting input from the user
  cout << "\nPlease enter a number: ";
  cin.getline( buffer, 32 );
  // compute final state
  STATE FINAL_STATE = Transition(buffer);
  // prints the final state
  PrintState(FINAL_STATE);
  return 0;
}
```

- Object code
  - 00101001011101010100101001000101010111 1

Recap: software development

- Modern software rarely written from scratch (80% of code writing is avoided - Daffara)
- Usually an assemblage of modules, with code gluing them together
- Methods of combining code/modules
  - Cutting and pasting
  - Linking (static/dynamic)
  - Plug-ins

Modern Software Development

- Code from many different sources is likely to mean one codebase contains many different copyright owners.
- Licensing structure can be complex
  - Licence/sub-licence
  - Parallel licences
  - Co-ownership

What is a licence?

Permission to do something that would otherwise be illegal.

That 'something' is usually copyright, but patent and trademarks are also relevant

Permission can be subject to conditions.

Cannot grant a licence to do something that is NOT illegal.

In common law countries, does not have to be a contract (usually is, in other countries)

Proprietary licences typically...

- Limit use to specified (number of) computers
- Restrict number of users
- Restrict types of use (e.g. home/student)
- Restrict jurisdiction
- Restrict assignment/transfer
- (Attempt to) restrict ability to reverse engineer
- Require payment of fees

Free/Open Source Licences Grant Freedoms

- Four Freedoms
  - The freedom to run the program, for any purpose (freedom 0).
  - The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.
  - The freedom to redistribute copies so you can help your neighbor (freedom 2).
  - The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

http://www.gnu.org/philosophy/free-sw.html

Open Source Initiative

1. **Free Redistribution**
2. **Source Code**
3. **Derived Works**
4. **Integrity of The Author's Source Code**
5. **No Discrimination Against Persons or Groups**
6. **No Discrimination Against Fields of Endeavor**
7. **Distribution of License**
8. **License Must Not Be Specific to a Product**
9. **License Must Not Restrict Other Software**
10. **License Must Be Technology-Neutral**

http://www.opensource.org/docs/osd

Licence Proliferation (2011)

1.	GNU General Public License (GPL) 2.0	45.64%
2.	Artistic License (Perl)	8.50%
3.	GNU Lesser General Public License (LGPL) 2.1	8.21%
4.	MIT License	7.94%
5.	GNU General Public License (GPL) 3.0	6.58%
6.	BSD License 2.0	6.25%
7.	Apache License 2.0	4.82%
8.	Code Project Open 1.02 License	2.52%
9.	Microsoft Public License (Ms-PL)	1.71%
10.	Mozilla Public License (MPL) 1.1	1.14%

http://www.blackducksoftware.com/oss/licenses/

Licence Proliferation (2012)

1.	GNU General Public License (GPL) 2.0	42.34%
2.	Artistic License (Perl)	7.94%
3.	GNU Lesser General Public License (LGPL) 2.1	7.08%
4.	MIT License	11.49%
5.	GNU General Public License (GPL) 3.0	6.41%
6.	BSD License 2.0	6.81%
7.	Apache License 2.0	5.48%
8.	Code Project Open 1.02 License	2.11%
9.	Microsoft Public License (Ms-PL)	1.88%
10.	Mozilla Public License (MPL) 1.1	1.02%

http://www.blackducksoftware.com/oss/licenses/

Licence Proliferation (2013)

1.	GNU General Public License (GPL) 2.0	32.65%
2.	Apache License 2.0	12.84%
3.	GNU General Public License (GPL) 3.0	11.62%
4.	MIT License	11.28%
5.	BSD License 2.0	6.83%
6.	Artistic License (Perl)	6.27%
7.	GNU Lesser General Public License (LGPL) 2.1	6.19%
8.	GNU Lesser General Public License (LGPL) 3.0	2.62%
9.	Eclipse Public License (EPL)	1.61%
10.	Code Project Open 1.02 License	1.33%

http://www.blackducksoftware.com/oss/licenses/

Types of client

Startups

SMEs

Transitioning software companies

SaaS companies

Multinationals

Startups

- Open source/open data friendly
  - Services model
  - Don't open source everything
  - Tend to be cloud based

SMEs

Difficult to categorise

Looking for second stage finance/exit

Need scalable income or intellectual property

VCs sceptical of open source – looking for IPR

Transitioning Software Companies

Obsessed with licence revenue

Difficult to transition to an open model

Try to 'hack' the open source model by

- sharing source, not allowing modification
- open core
- dual licensing (using licences like AGPL)
- minimal compliance with GPL
- use 'open' as marketing, not philosophy
-time release software (Ghostscript)

Danger signs

Use words like 'viral', 'cancer' and 'infection' when talking about GPL

Talk about 'freeware'

Put a lot of effort into engineering shims etc. to avoid the GPL

Dual license using AGPL

Talk a lot about open source but put everything in the cloud

SaaS Companies

Can be good or bad citizens

Are they abusing the spirit of open source?

[Open APIs – or not?]

Need to remember they are likely to be distributing code on the client side (e.g. javascript)

Multinationals

Patent focussed

Scared of GPL, or ignore it

Can be good corporate citizens

Clients' view on s/w patents

- Most SMEs:
  - Ignore them and hope they go away
- Thinking SMEs
  - Patents are too expensive to apply for
  - Patents take too long to apply for/grant
  - Copyright protection works for us
- Larger companies with an existing patent portfolio
  - Embrace patent model
  - Useful to counter incoming infringement suits
  - Are a “fact of life”
- Rare exception – image processing developer
  - The value is in my technique, not the code

Open Source Companies

- SMEs
  - Oppose patents
  - May join patent pools for their defensive capability
- Larger Companies with existing patent portfolios
  - More wary – want to retain value in existing portfolio
  - May out-license patents for open source projects on various bases

The Licences

- “Ignore it and hope it goes away”
  - MIT
  - BSD
- Licences with explicit patent clauses
  - Apache
  - Open Software License (Academic Free License)
  - Mozilla
- Liberty or death!
  - GPL

Context

An open source license must grant enough patent rights to allow the licensee to make, use, sell, offer for sale, or import the open source work as distributed by its licensor. Any additional license rights for derivative works or other uses are at the option of the licensor.

Larry Rosen

The MIT License

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Apache 2.0

An academic licence, but:

1. Longer

2.Patent license

3. Patent retaliation (covering patent license only)

Mozilla 2.0

Weak copyleft, which

1. Allows the executable to be released under any licence...
2. ...but requires the source to be released under this licence
3. Covers the file only (not the project)
4. Relicensable in GPL2+, LGPL2.1+, AGPL (optional)
5. Patent licence, retaliation covers whole contributor version

GPL 2.0

Strong copyleft

1. Requires all 'works based on the program' to be released under GPL.
2. No explicit patent licence
3. 'Liberty or death'
4. Any third party receiving the binary must be able to access the source
5. No additional restrictions

GPL3

Introduced in 2007 after a long consultation

- TiVoisation
- Licence compatibility
- Microsoft/Novell deal (and patents generally)
-Flexibility on additional terms
-Internationalisation (convey/propagate)
-Bit torrent
-Permission to circumvent TPMs
-Clarity on curing violations

GPL 3

- Extremely complex patent clause
- Grants rights for essential patent claims controlled by a contributor, only covering the contributor version.
- Addresses deals like Microsoft/Novell deal
- Liberty or Death!

Liberty or Death! GPL3

- Or: “no surrendering others' freedom”

  If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

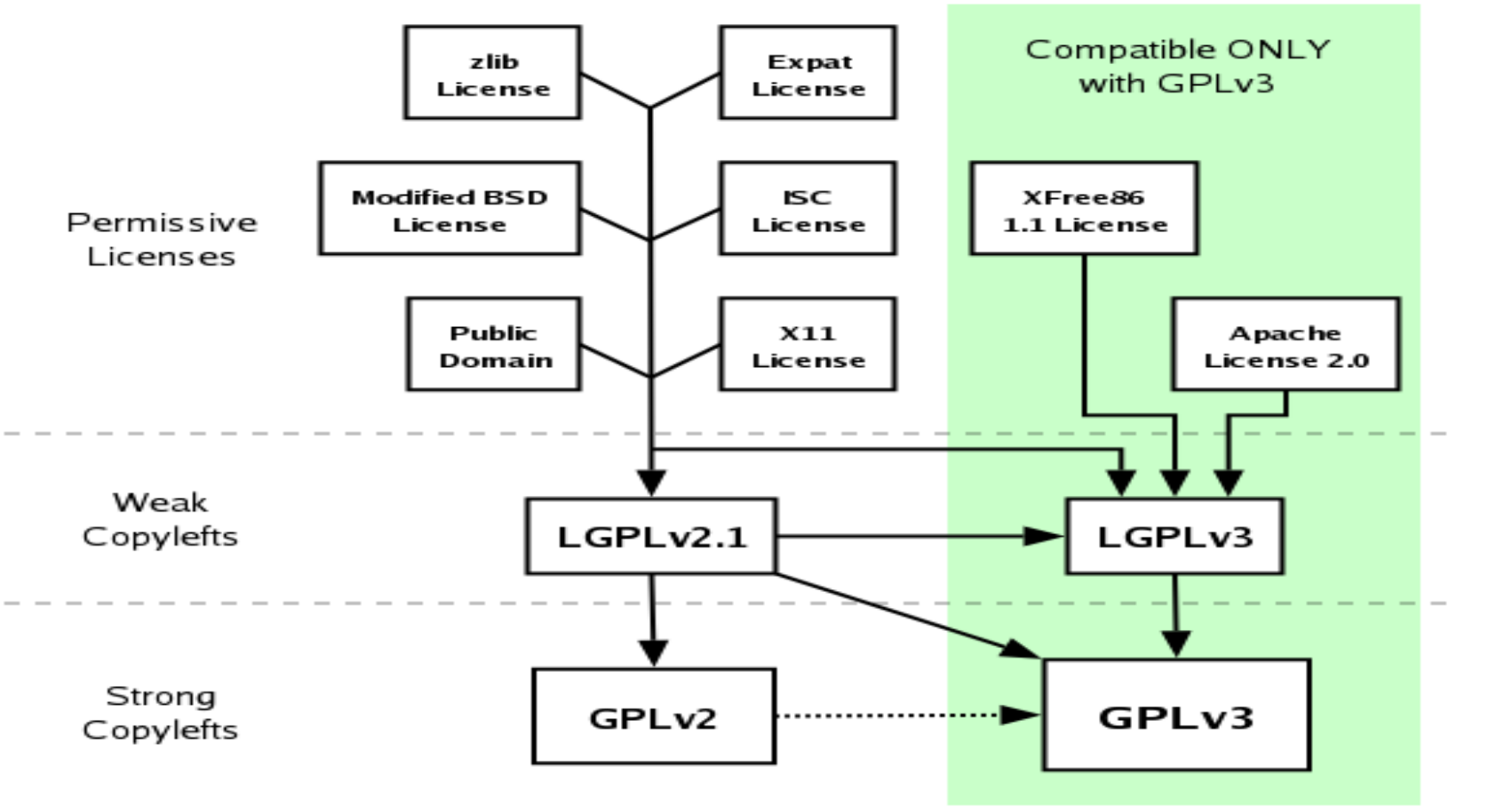
Projects and Licence Compatibility

- Do you want/need to out license at all?
- If so, what out-licence do you want to use?
  - Proprietary
  - Copyleft
- Need to ensure that in-licences are compatible:
  - Academic licences are generally not a problem
  - Copyleft licences usually require outlicensing under the same licence, except:
  - E.g. GPL2/3 not compatible, but
    - “or later” clause can help
    - EUPL allows relicensing under GPL, CeCILL
    - Mozilla 2.0 allows relicensing to GPL, AGPL, LGPL (unless excluded)

Compatibility and Linking - GPL

- If definitely a derivative work, will be a compatibility problem
- If on borderline (e.g. dynamic linking to a GPL library), then may well annoy the community
  - E.g. Wordpress themes
- Technical ways of circumventing:
  - Separate download, shims
  - May also annoy the community, even if legally permissible

GPL Compatibility Chart (fsf.org)



The Questions

Open Source license vs. proprietary license

Permissive vs. copyleft licenses

Safe and 'dangerous' Open Source licenses

Obligations related to use of GPL

Restricting redistribution of Open Source solutions

Reasons for obtaining copyright of the code

Changing Open Source license

Use of mixed licenses

Governance of Open Source projects and its relationship with Open Source licensing

Potential license mismatch when using GPLv2 & LGPLv3

The MPLv2 vs. LGPL

Dealing with third party contributions

Mixing Open Source licenses

Indemnification of safety-critical software with a GPL v3 license -- what is the business model? Is it possible?

Open Source vs Proprietary Licences

What is really an open source license? I have heard about the Open Source Initiative, but how does it work in practice? Can I invent a new license and call it an Open Source license? What are the key difference between different licenses?

When we now enter the cloud, why should anyone care about software licenses?

Permissive vs. copyleft licenses...

What are the key differences between permissive, weak copyleft, and strong copyleft Open Source licenses? If I am an IT manager in a public sector organisation:

What are the main reasons for using (and not using) a permissive Open Source license?

What are the reasons for using (and not using) a weak copyleft Open Source license?

What are the reasons for using (and not using) a strong copyleft Open Source license?

Would the recommendations be different if I am a CEO of a small IT-company wishing to adopt Open Source software components that are available and incorporate those in some software that we use internally. Would the recommendation be different if we also use and distribute the software to external organisations (customers)?

Would the recommendation be different if I am a manager in a large company wishing to use and distribute software that we develop for our customers?

Safe and 'dangerous' Open Source licenses...

Are there any specific Open Source licenses one should avoid when distributing software that has been developed internally in our company?

Are there any specific Open Source licenses one should consider using when distributing software that has been developed internally in our company?

Some large companies to not want to engage with GPL-projects. Why? Are these views based on misconceptions or are there something inherently problematic from the perspective of a large company?

Obligations related to use of GPL...

When our organisation distribute a binary which contains GPL licensed software, what must we do in addition to providing our customer with the binary? What should we do (in addition to what we, from the licensing point of view, have to do)?

Restricting redistribution of Open Source solutions...

Suppose our organisation (we are an IT consultant company) have developed a software system for one of our customers. We have now developed and distributed the software system to our customer and they have started to use the software. One aspect of the agreement with our customer is that the software should be licensed under an Open Source license. Suppose we license the software system under a copyleft license (e.g. GPLv2).

Under what circumstances can our customer prohibit us from distributing the software to another organisation? For example, suppose the contract with our customer contains a clause which states that redistribution of the software system to other organisations is not allowed. Would such a clause in a business contract be a violation of the underlying Open Source software license agreement? Would there be a difference if the software system is licensed under a permissive license or a weak copyleft license (e.g. LGPLv2)?

Similarly, if we consider the inverse situation (i.e. if I work as an IT manager in a public sector organisation and are a customer of the IT consultant company), under what circumstances can our supplier prohibit us from redistributing the software to another organisation (e.g. another public sector organisation have asked us if they can get a copy of the software we have)? Would there be a difference in this situation, if the contract with our supplier contains a clause which states that redistribution of the software system to other organisations are not allowed (i.e. our supplier insisted on keeping the copyright and also that we should not be allowed to redistribute the solution). Would such a clause in a business contract be a violation of the underlying Open Source software license agreement? Would there be a difference if the software system is licensed under a permissive license or a weak copyleft license (e.g. LGPLv2)?

Reasons for obtaining copyright of the code...

Suppose I represent a public sector organisation, under what circumstances would it make sense to a ask a supplier to obtain the copyright of the code? Would it make sense to have a shared copyright of the code?

Changing Open Source license...

Suppose I encounter an Open Source software project and I really like the code. However, I do not like the license under which it is provided. What can (and should) I do in order to be allowed to use the code? Under what circumstances can the license be changed?

Use of mixed licences

We sometimes see the AGPL license used in combination with a proprietary license (e.g. status.net), and sometimes GPL is used in combination with a proprietary license (e.g. MySQL). What are the reasons for using (and not using) a mixed strategy?

Why are some Open Source projects provided under several different Open Source licenses? What are the main motivations for this?

The MPLv2 vs LGPL

he MPLv2 license has received some attention and been adopted by governmental agencies

(e.g. a Danish governmental initiative used this for a PDF-project). Some consider this license to be a viable alternative to the LGPL license. What are the key differences between MPLv2 and LGPLv2? What effects that we currently get from LGPLv2 would we loose if we were to change the license for our software (to MPLv2) instead? As community members (and contributors) to GPL-licensed Open Source projects implementing the PDF-file format express some concern related to patents that Adobe have on the PDF file format (as standardised by ISO) would you recommend using LGPLv3 instead as a strategy for ensuring continued openness of the Open Source project (and as a strategy for ensuring that the licensing conditions for the PDF-specification (which is current RF-conditions) will not change related to future versions of the PDF file format (as standardised by ISO).

In general, if we were to start a new project today for implementation of a file format and are considering MPLv2 and LGPLv3 as alternatives, which one of these two alternatives would you recommend? What effects would we loose if we were to use LGPLv2 (instead of LGPLv3)?

Governance of Open Source projects and its relationship with Open Source licensing...

Governance of open source projects under foundations has been claimed to be an effective way of creating a "legal shell" around such projects in order to avoid lawsuits stemming from license or patent infringements. Do you think this is a "bullet proof" strategy as a protection against legal problems or are there weaknesses or limitations with this approach to governance?

Related to this, what is the role for foundations?

Potential licence mismatch when using GPLv2 & LGPLv3...

Assume that one wants to publish a software package denoted by NEW and that this software package is dependent on another package X which in turn is dependent on Y which in turn is dependent on Z. Further assume that the licenses for each of X, Y, Z are the following:

X is licensed under GPL 2 or later
Y is licensed under GPL 2 or later
Z is licensed under LGPL 3 or later

Since X and Y is dependent on Z, which is licensed under LGPL 3, is there a license mismatch? If there is a license mismatch and since program NEW is dependent on X, what should one do as an author of NEW? If there is a license mismatch, should the authors of X and Y change the license to, e.g., GPL 3 or later?

Mixing Open Source licenses in a distribution package...

Is there a recommended way, or are there any restrictions how to deal with mixed licenses for different files in a distribution package? A concrete example: EJBCA and SignServer are both JEE applications where the (java) code is under LGPL v2.1 (or later). All contributed Java code must be under LGPL 2.1 or later, anything else would not work, right? We sometimes get helper/utility scripts contributes, where the header says GPLv3 (or later). The scripts are not needed to run or use the applications, but are nice additions and should be included in the distribution zip file.
Is there any recommendations for such cases? Is there a license for the "whole work"?
- Can we claim the "distribution" is LGPLv2.1 (or later), with the helper script being GPLv3?
- Is there any such thing as a license for the distribution?
- Do we need to list all licenses for all helper scripts, or libraries we depend on, in the "distribution" license README?

Dealing with third party contributions... Dealing with copyright assignments...

Are there any recommended practices for third party contributions?

- Sign off from contributors "this is my original work, not copied from something else"?
- Copyright assignments? Why (and when) should one use such? Why (and when) should one not use such?
- License headers? (for whole files this or easy, but for some lines for an existing file it is not)

Indemnification of safety-critical software with a GPL v3 license

What is the business model?

Is it possible?

Further Resources

- IFOSSLR www.ifosslr.org
  - Issue 1 – The Risk Grid
  - Jacobsen –v- Katzer
- www.fsf.org
- www.rosenlaw.com
- www.groklaw.net
- www.w3c.org
- www.opensource.org