IP and Licensing

Nigel Goddard

School of Informatics
University of Edinburgh

Once upon a time

nobody thought that 'intellectual property' was important in software.

Times changed, and software fell into the embrace of the IP lawyers.

Intellectual Property

IP is not property, and it's not intellectual. It resists definition, but one definition of IP might be: a monopoly right to exploit an intangible product of human thought or labour.

The concept developed from the 16th century. Best-selling authors resented other printers printing their books and selling them cheaper with no payment. The notion of copyright – the monopoly right to publish copies of a book – developed, and through repeated lobbying was enshrined in law and extended – and extended – and extended.

Modern intellectual property rights

fall into several broad classes.

- Copyright applies to literary or artistic works.
- Patents apply to inventions of things or processes.
- Design rights apply to the design of products.

None of these were invented with software in mind...

Patents

arose to protect inventors of physical objects or processes (e.g. steam engine).

Stronger than copyright: stops other people using/making object, even if they invented it independently.

Duration is shorter – 17 years.

Controversy over what is patentable. Originally, physical things and processes for making physical things.

Extended to 'embodied programs'. Can an algorithm be patented? In U.S., yes. In Europe, no, roughly speaking.

In U.S., 'business processes' are patentable. E.g. Amazon has patents on 'one-click' shopping, and on the idea of customers reviewing products!

Patent systems appear unable to cope with the issues of IT patents.

Copyright

is the original IP.

- Restricts the ability to copy, adapt, etc. artistic or literary works.
- But no artistic or literary merit required. This lecture is a literary work for copyright purposes.
- Currently subsists (on literature) for life of author plus 70 years.
- The protected rights may be assigned in whole or in part, or licensed in whole or in part, with or without restrictions. (Certain exceptions for moral rights.)

Generally, it is accepted that source code is subject to copyright.

Object code and machine code are 'adaptations or translations' of source code, and hence protected.

Running a program necessarily involves copying it!

Scope of copyright

Slogan: copyright protects the expression of an idea, not the idea itself. Expression includes, e.g., cricket scoring charts.

Largely by judicial interpretation, this has been strengthened. Characters and settings, plots and stories, have been held to be copyrightable.

What of reverse engineering? Generally held that clean-room reverse engineering does not breach copyright – the function is not protected, the code is.

Almost all code (compiled or not) is licensed to the user within the framework of copyright.

Standard commercial licences

Most commercial software is licensed either per computer or per user. License grants permission to run program; usually specifically forbids decompilation etc.

Per computer: usual model for desktop PC software. Easy to arrange, check, and charge for.

Per user: usual model for more expensive software, particularly SE tools, scientific tools, etc. Licences may be tied to individuals, or may 'float', using a licence server to control total number in use. (E.g. Matlab.)

Shareware

Common for small cheap PC applications. Demo version freely available, but limited in functionality or lifetime. Purchase of a key unlocks full version.

Free software / Open source etc.

Numerous rather doctrinal arguments about terminology. Often now called floss (Free/Libre Open Source Software).

Open source: the source code must be available and open for review *and* modification by users. No charge may be made for licence. Redistribution (even of modified versions) must be allowed.

For an attempt to pin down the meaning of 'open source', see (required reading)
`http://www.opensource.org/docs/definition.php`

There are many widely used licences: the BSD Licence, the X11 (MIT) Licence, the Mozilla Public Licence, the Artistic Licence. We'll look at two Gnu licenses: GPL and LGPL.

You should be familiar with the difference between GPL and LGPL and when it is appropriate to use which one.

Gnu I: The General Public Licence

The GPL goes beyond earlier 'free' licences. It was the first *copyleft* license – key property that modifications or adaptations of GPL'ed software, *including* software using any GPL libraries, must be licensed under the GPL. ('The viral nature of the GPL.')

Note that the GPL does not force you to distribute your modifications; and does not prevent you charging fees for providing your software; and does not prevent you charging for support.

The GPL is incompatible with many other licences; it is legally impossible to combine incompatible software with GPL'ed software.

Appropriate when you want your software to be freely accessible, and you don't want anyone who keeps their software closed to be able to use yours.

Required reading:
`http://www.fsf.org/licensing/licenses/gpl.html`

GNU II: The Lesser General Public Licence

The LGPL places copyleft restrictions on code, but does not apply these restrictions to other software that merely link with the code.

So proprietary applications can link with LGPL'ed code and not have to be distributed freely, even though the LGPL'ed code is distributed freely.

The LGPL is mostly used for software libraries, but also some applications e.g., Mozilla and LibreOffice.

Appropriate when you want your software to be freely accessible, but you are happy for it to be included in software that is not freely accessible.

Required reading:
`http://www.fsf.org/licensing/licenses/lgpl.html`

The choice of licensing model

The range of models varies from 'keep it secret and charge the earth' to 'give it away'. The choice depends on many things:

- Philosophy: some consider restricting software to be unethical.
- Legal constraints: you may have used other software that restricts your choices.
- Business relevance: is the software your core business, or do you use it to leverage other (chargeable) activities?
- Support: do you want to get the users to contribute?

Reading

Required:

`http://www.fsf.org/licensing/licenses/gpl.html`

Required:

`http://www.fsf.org/licensing/licenses/lgpl.html`

Required:

`http://www.opensource.org/docs/definition.php`

Quote of the day

> *People generally seem to want software to be free as in speech and/or free as in beer. Unfortunately rather too much of it is free as in jazz.*

Janet McKnight, in uk.misc.