Richard Stallman's

“The Danger of Software Patents”

FREE SOFTWARE FOUNDATION

Copyright, Patents, Trademarks

Origins are independent.

Public policy issues raised are different.

"Intellectual Property" is biased.

“Intellectual Property” leads to simplistic conclusions.

Differences between copyrights and patents

- Copyright deals with details of work, not ideas
- Patents cover ideas and use of ideas

- Copyrights happen automatically
- Patents issued by a patent office

- Patents cost lot of money
- Copyrights are owned by authors

- Copyrights last long, for example, 150 years
- Patents last 20 years

- Copyrights cover copying
- Patents are a monopoly on the use of an idea

Drawbacks of Patents

Patents publishing time is around 18 months.

Pending patents are kept secret.

Your program might be released before a patent on the program is published.

Threat of getting sued.

Example: LZW compression algorithm in compress command, 1984; patent released in 1985.

Drawbacks of Patents (II)

Thousands and thousands of patents.

Patents are ambigously written.

Example:
Patent on "Natural order recalculation" in "spreadsheets" in 1963;

uses the "toplogical sort algorithm"; none of the above keywords are mentioned in the patent;

only describes a method of "compiling formulas into object code".

Drawbacks of Patents (III)

Patents are hard to understand.

Australian government made a study on the patent system in 1980.

Conclusion: "aside from international pressure, there was no reason to have a patent system".

One of the engineers said: "I can't recognize my own invents in patents".

Drawbacks of Patents (IV)

Example: Around 1990, Paul Heckel sued Apple claiming that hypercard infringed his patents;

Lawyer told him that he could read the patents as coverying hypercard;

RMS gave a speech at Stanford and quoted this example. Paul Heckel, who was in the audience replied, "That's not true, I just didn't understand the extent of my protection!". RMS, replied saying, "Yes, that's what I said".

Business perspective

Three ways of handling patents:

Avoiding patents

Licensing the patent

Overturning the patent in court

Avoiding patents

Easy or hard depending on the idea.

If a feature is patented, you can avoid implementing the feature.

Example: XyWrite word processor provided a downgrade for users to remove pre-defined abbreviation feature.

Ideas may be too broad or on an entire field.

Example: Public Key Encryption was patented in US; expired in 1997.

Avoiding patents: Examples

Optimized version of FFT (twice as fast as ordinary FFT); you can use ordinary FFT instead.

gzip compression algorithm was not patented; now de-facto standard for data compression.

Pantone color matching not in GIMP, available in Adobe Photoshop; feature patented by Pantone.

Avoiding patents: Patented twice

Two patents covered LZW compression algorithm.

Patent office doesn't have the time.

US patent office spends on an average 17 hours per patent.

Software is pure mathematical and a single calculation can be described in many ways.

Avoiding patents: Patents versus Standards

Company or consortium can make a format/protocol a de-facto standard.

Standards exist that might have patents on them.

Standards have to be free for anyone to implement.

Avoiding patents:
Patents versus Standards (II)

Example: WWW consortium proposed to start adopting standards covered by patents.

Community objected and they reverted themselves.

Some standards committees refuse to issue a standard which they KNOW is patented. However, it could be covered by a patent they don't know about.

The standards committee adopts the standard, thinking it is not patented--and afterwards, some patent holder comes out of the woodwork and attacks the users. That is what happened with JPG.

Licensing the patent

Patent holder does not need to offer a license.

Example: A family business using gambling machinery for casinos that used computers;

The patent covered "having a number of computers on a network for playing games such that each computer supported more than one game and allowed you to play more than one game at a time";

Patent holder didn't offer license and the business had to be shutdown.

Licensing the patent

Licenses are charged.

Natural order recalculation patent demanded 5% of gross sales of every spreadsheet in the US.

Question arises if you have to pay for 20 different patents.

2-3 patents with 5% for licensing are sufficient to make any business unfeasable.

Licensing patents: Suitable for big players

Licensing patents is a very good solution for multinational mega-corporations because they own lots of patents.

Big companies cross-license with each other.

Example: IBM article in Think Magazine (No. 5, 1990) which said they have two benefits from their 9000 US patents:

a. royalty
b. getting access to patents of others

The benefit of the latter being 10 times greater than the former.

Licensing patents:
Suitable for big players (II)

This phenomenon of cross-licensing refutes the common myth that patents "protect" the "small inventor".

Big corporations benefit from them and hence push for software patents.

Overturning a patent

Patents are given to the obvious.

Costs lot of money to fight in the court.

Example: In one patent case, Qualcomm invested 13 million US dollars of which most went to lawyers.

Patents are there in other fields? Why not software?

It is like saying "Some people get cancer. Why should you be exempt?"

Patents relate differently to different fields.

In pharmaceuticals, a given chemical formula would be patented, so, there is one patent per product and it covers the idea of the product.

Patents are there in other fields? Why not software? (II)

Software packages are big, complex, and cover thousands of ideas. So, there are thousands of points of vulnerability in your program.

Software patents thus tend to obstruct the development of software.

Software patents do not allow invention or innovation.

Why software patents cannot be applied to software

Software consists of ideal mathematical objects and can be implemented in many ways.

Other fields deal with physical matter.

Software doesn't have to worry about the physical boundary limitations.

Software systems are huge and far complex than physical systems.

Analogy : Symphonies

A symphony is long and has many notes in it, and uses many musical ideas.

One may have a lot of new musical ideas but he/she has to use a lot of existing musical ideas in order to make recognizable music.

Analogy : Symphonies (II)

Music, like software (and writing and in fact all other fields of human endeavour) builds continuously on its own past. You cannot write music or software entirely out of new ideas.

Nobody is so brilliant that he can re-invent music and make something that people would want to listen to; and the same is applicable to software.

Publish ideas

In earlier days, people published ideas and got credit for them, but often kept the code a secret (unfortunately).

Today, people patent the ideas and still keep the code secret.

Patents have not encouraged disclosure.

Policies in countries

Totally remove the use of software patents.

Question the patents issued by the patent office.

All software developers are threatened by software patents, and even software users.

WTO follows corporate regulated trade, as opposed to free trade.

In Europe, software patents were blocked twice.

Links

http://www.researchinnovation.org

http://www.programming-freedom.org

http://www.ffii.org

http://www.gnu.org/philosophy/stallman-mec-india.html

http://swpat.ffii.org/

Thank
You