The Plague of Software Patents

DU Pirate Party

January 29, 2013

Introduction

These slides are only a rough skeleton for the talk! Several of the points need elaboration.

These slides are largely based on content at swpat.org.

Outline:

1. What are software patents?
2. State of affairs in Europe
3. Trivial patents that have been granted
4. Arguments against software patents

“Method”

- Software patents protect a “method” not an executable program (its “expression”)
- What exactly is a “method”?
- How do we conceptualize or model what constitutes a “method”?

Some "methods"

Patent Applications (first four have been granted in the US)

- 'Method and system for stacking toolbars in a computer display' (Microsoft, 1995)
- 'Method and system for identifying and obtaining computer software from a remote computer' (Microsoft, 1995)
- 'Method and system for mapping between logical data and physical data' (Microsoft, 2004)
- 'Method and system for downloading updates for software installation' (Microsoft, 2002)
- 'Method and system for scaling a graphical user interface (GUI) widget based on selection pointer proximity' (IBM, 2001)

European Patent Convention: Article 52 [1]

1. European patents shall be granted for any inventions, in all fields of technology, provided that they are **new**, involve an **inventive step** and are susceptible of industrial application.
2. The following in particular shall not be regarded as inventions within the meaning of paragraph 1.
    1. discoveries, scientific theories and mathematical methods;
    2. aesthetic creations;
    3. schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computer
    4. presentations of information.
3. Paragraph 2 shall exclude the patentability of the subject-matter or activities referred to therein only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such.

[1] from http://www.epo.org/law-practice/legal-texts/html/epc/2010/e/ar52.html

Arguments Against Software Patents (I)

- **Impediment to development:**
  Prevents parties from using particular techniques or solutions.
  - Free and open source project developers face the threat of litigation - sometimes a project is shutdown or fragmented.
  - Stifles innovation and development of small teams and individuals.
  - Leads to the adoption of potentially less optimal solutions $\Rightarrow$ possible loss of performance, functionality and compatibility.

    Example: Paul Rubin observed that one company chose one patented product over another for "litigation insurance rather than technology".

  - Discourages some organizations from releasing Free and Open Source software.
  - Results in much effort being expended to create open patent-free alternatives (think Vorbis). Consumes the time of communities.
  - and many other reasons...

Example: Elliptic Curve Algorithms

- Certicom hold several patents relating to Elliptic Curve Cryptography (ECC). Many of these involve optimizations to the *implementation* of ECC schemes.
- RIM is another holder of many ECC patents.
- Lack of clarity about what is and what is not covered in these patent portfolios and others.
- Fear of patent infringement has contributed to the slow uptake of ECC.
- The RSA patent has expired but several ECC patents remain alive!
- **Many Free and Open Source ECC libraries available - good performance + unencumbered by patents. But we are still denied a range of some good techniques.**

Arguments Against Software Patents (II)

- **Obviousness and Triviality**
  - Given a problem context, intuitive and natural solutions should not be patentable.
  - Programmers tend to often independently "converge upon" a suitable algorithmic approach to achieve a task.
  - There are countless patents containing *obvious* algorithmic solutions and trivial technical content.

Example: Doubly Linked List

- Patent issued in the US in 2006 for:
  'Linked List' (Covers doubly linked list!) (LSI Logic Corporation, **2006**)
- See 'The Art of Computer Programming' by Donald Knuth; **Volume 3, first edition, 1973**!
- How much software has been written that uses a doubly linked list before this patent was granted?

Why more trivial patents in the area of software?

Why are obvious and trivial patents granted more often in the area of software? Some possible reasons for this:

1. Software is comparatively new and trends change fast.
2. Barriers to "invention" are lower than other subject areas e.g. electronics, pharmaceuticals, ... Easier to troll.
3. Many software patents pursue a new niche problem and provide an obvious solution for it. Examiners conflate the obviousness of the latter with the lack of prior art for the former since the particular problem configuration in question may not have surfaced elsewhere yet.

So what about campaigning exclusively to improve the assessment standards?

- This may reduce the quantity of trivial software patents but it fails to solve the fundamental problems (see [1]).

A Plethora of Other Arguments

- Arguments abound from different perspectives.
- An interesting list has formed at [1] (`http://en.swpat.org/wiki/Why_abolish_software_patents`).

A Succinct Argument [2]

Connection between software and mathematics

"I am told that the courts are trying to make a distinction between mathematical algorithms and nonmathematical algorithms. To a computer scientist, this makes no sense, because every algorithm is as mathematical as anything could be. An algorithm is an abstract concept unrelated to physical laws of the universe."

Donald Knuth

[2] See [2]

A Succinct Argument (II)

Roughly speaking:

1. The subject of a software patent is an "effective method".
2. By the Church-Turing Thesis, any "effective method" can be computed by a Turing Machine or an equivalent model of computation.
3. Therefore, the "method" presented may be exhibited as a well-defined mathematical formula.
4. Mathematical formulae are abstract concepts.
5. Abstract concepts cannot be patented (e.g. p2, Article 52 of the EPC)

Can be argued in other ways as well e.g: we can invoke the Curry-Howard Isomorphism.
Can this be extended beyond software?
What are the subtleties in the above sequence?

Finally...

- This talk has only scraped the surface.
- Recommendation: the short documentary 'Patent Absurdity' supported by the FSF - available at http://patentabsurdity.com/download.html
- Recent lawsuits shed more light on this broken system.
- Check out endsoftpatents.org.
- Thank you for your interest.

swpat.org, “Raising examination standards wouldn’t fix much,” 2013. [Online]. Available: http://en.swpat.org/wiki/Raising_examination_standards_won%27t_fix_much

——, “Software is math,” 2013. [Online]. Available: http://en.swpat.org/wiki/Software_is_math