Patents & Freedom: Where We Stand Now

McCoy Smith
Lex Pan Law
Opsequio

opsequio
Virtual Open Source Program Office Services

<all caps>Mandatory Disclaimer</all caps>

- IAAL
  - IAAAPL
    - BIAN*Y*L
  - For your legal questions, ask your lawyer!
    - They may have a different opinion
- I have simplified concepts, opinions, etc. for ease of presentation
  - Please, no quibbling, lest I quibble in return
    - Please hold questions for the end

Identify this Quote

- “[E]very program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary.”

Identify this Quote

- “[E]very program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary.”
  - GPLv3 (2007)
    - GPLv2 (1991) has similar language

What is Patentable?

- “Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor”
  - 35 USC § 101
- The U.S. Supreme Court has stated there are at least 3 specific exceptions to the broad rule above:
  - laws of nature
  - physical phenomena
  - ***abstract ideas***

Abstract Ideas

- For almost 40 years, the question of to what extent particular software innovations are “abstract ideas” has been the subject of much debate, inside and outside the courts

A Brief History of Software & Patents

- First U.S. Patent Applications on software filed in the mid-1960s
  - Example: US Patent No. 3,380,029
- First US Supreme Court case examining a software-related patent found it to be not eligible as merely an “abstract idea”
  - Gottschalk v Benson (1972)
    - Technique for converting binary coded decimal numbers to pure binary

A Brief History of Software & Patents

- US Supreme Court validates granting of software-related patent in the early 1980s
  - Diamond v Diehr (1981)
    - Computer + known mathematical formula + new machine configuration to gather data to feed into that computer that processes that formula is no longer an “abstract idea”

A Brief History of Software & Patents

- Early ‘10s produces more Supreme Court guidance on eligible invention vs abstract idea
  - Bilski v Kappos (2010)
    - Computer method for optimizing energy trading unpatentable as abstract idea
      - But:
        - “Business method” software patents not per se ineligible
        - Software, not tied a specific machine, not per se ineligible
        - Patent law allows patents on “processes”; that could include software

A Brief History of Software & Patents

- Early ‘10s produces more Supreme Court guidance on eligible invention vs abstract idea
  - Alice v CLS Bank (2014)
    - Software-managed financial trading system to reduce certain risks is ineligible “abstract idea” (i.e., another business method patent)
    - But:
      - An “abstract ideas” with an added “inventive element” may be patentable
    - FSF filed amicus brief arguing patentable software must either:
      - Be used by a specific machine (not general purpose computer) or
      - Physical transformation of an “article” from one state to another

A Brief History of Software & Patents

- Late ‘10s produce lower appeal courts struggling with Supreme Court guidance on eligible invention vs abstract idea
  - Enfish v Microsoft (Federal Circuit 2016)
    - “Self-referential” relational database
    - Found valid as non-abstract
  - TLI Communications LLC v AV Automotive, L.L.C. (Federal Circuit 2016)
    - Assigning classification data digital images, sending images to a server, storing on server taking into consideration the classification information
    - Found invalid as abstract
- Are those decisions reconcilable?

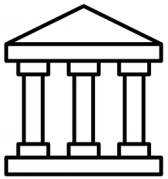
What does this all mean?

- **Courts** (at least some of them) have a degree of skepticism of the sorts of software-related patents filed within the past 20 years
  - i.e., the ones still in force, particularly non-technical ones
  - Lots of broad claims to general techniques using computers, without a lot more
    - Tend to be found too abstract
- But other software developments, if more specifically detailed and carefully written, will survive the “abstract” test
  - Good patent lawyers hedge bets by using different claiming style

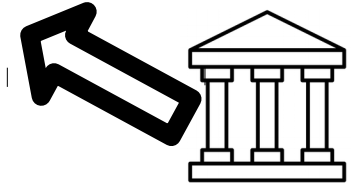
What does this all mean?

- **US Patent Office** may be granting high % of patents “related to” software
  - From 34% in 1991 to 61% in 2019?
    - https://www.ipwatchdog.com/2019/07/02/congress-contemplates-curbing-alice-60-issued-u-s-patents-software-related/id=110920/
    - What’s a “software patent” not exact science
    - Take those numbers with a grain of salt
  - Uptick probably reflects
    - Skilled patent lawyers “writing around Alice”
    - “Software eating the world”
    - Increasing integration of software into more and more products

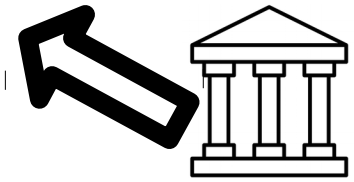
Where Patents Are Evaluated



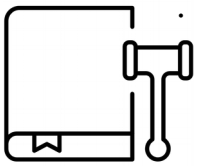
- US Supreme Court
  - Patent cases rare
    - Demonstrated skepticism of software patents
      - Interpretive rules can be quite generic



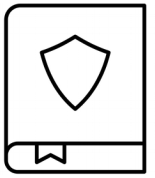
- Court of Appeals for the Federal Circuit (“Supreme Court of Patents”)
  - Patent cases common
    - Less skepticism of software patents?



- US District Courts
  - Skepticism depends on location
    - Beware Texas; Bay Area better?



- US Patent Trial and Appeal Board
  - Skepticism of **all** patents?
    - “Patent Death Squad”



- US Patent Office
  - Less skepticism of software patents
    - Guided by Supreme Court decisions

What about ROW?

- European Patent Office, Guidelines for Examination, 3.6:
  - Computer programs are excluded from patentability ... if claimed as such. … [T]he exclusion does not apply to computer programs having a technical character. … [A] computer program must produce a "further technical effect" when run on a computer. A "further technical effect" is ***a technical effect going beyond the "normal" physical interactions between the program (software) and the computer (hardware) on which it is run***. ...***Examples of further technical effects*** which confer technical character to a computer program ***are the control of a technical process or of the internal functioning of the computer itself or its interfaces***.

What about ROW?

- European Patent Office, Guidelines for Examination, 3.6:
  - [T]he mere fact that a computer program serving a non-technical purpose requires less computing time than a prior-art program serving the same ... purpose does not on its own establish the presence of a further technical effect. … ***[C]omparing a computer program with how a human being would perform the same task is not a suitable basis for assessing if the computer program has a technical character.*** If a further technical effect of the computer program has already been established, the computational efficiency of an algorithm affecting the established technical effect contributes to the technical character of the invention and thus to inventive step (e.g. where the design of the algorithm is motivated by technical considerations of the internal functioning of the computer).

What about ROW?

- India, New Zealand: anti-software patent
  - But not completely anti
- PRC: pro-software patent
  - Law changed in 2017 to be more Europe-like
    - Previously generally anti-software patent
    - PRC currently advertising itself as more patent-friendly than U.S.A.

How Patents Are Challenged

- How software patents are challenged:
  - Eligibility – 35 U.S.C. § 101 – i.e., the “abstract” question
  - Novelty – 35 U.S.C. § 102 – is prior art identical to patent claim?
  - Obviousness/Inventive Step – 35 U.S.C. § 103 – is there prior art not identical to the patent claim, but that claim is an obvious extension of or improvement upon it?
- Prior art is by far the most common mechanism to challenge patents
  - And is a requirement institute one type of external challenge (IPR) through the US Patent Office

Where & When Patents Can Be Challenged

- U.S. District Courts
  - There needs be some sort of real threat by patent holder against challenger to start
  - Doesn’t require prior art to challenge
  - Very expensive, and likely results in responding infringement claim

Where & When Patents Can Be Challenged

- Patent Trial and Appeal Board (PTAB)
  - IPR (Inter Partes Review):
    - Must be based on prior art
    - If there is already litigation between patent holder and requester, must file request within a year
    - Semi-expensive (filing fees $15K, legal fees much more)
  - Post-Grant review
    - Can be based on any validity problem (including unpatentably “abstract”)
    - Must be filed within 9 months of patent issue
    - Semi-expensive (filing fees $15K, legal fees much more)

What Can I, A Non-Lawyer Do?

¯_(ツ)_/¯

Linus's Law

- "Given enough eyeballs, all bugs are shallow"

The Patent Corollaries

- “Given enough free software, prior art is shallow”
- “Given enough free software developers – and free software-friendly patent lawyers – patent design-around is shallow”

Patent Corollary #1

- “Given enough free software, prior art is shallow”
  - Good prior art against bad patents must be:
    - Date-verifiable
    - Publicly available
    - Findable
      - Biggest need here: documentation susceptible to natural-language searching and identification
      - Source code alone may not be sufficient

Patent Corollary #2

- “Given enough free software developers – and free software-friendly patent lawyers – patent design-around is shallow”
  - Think of any patent as a bug, that can be coded around
  - Design-around is a two-step process
    - Analyze patent record to understand what is and isn’t within a patent’s legal rights (patent lawyers)
    - Rewrite code to fall outside the patent’s legal rights (free software developers)

What Can I, A Non-Lawyer Do?

- Write, and publish, free software
  - Document, document, document
- If need be, be ready to help do code-around for patent bugs
- Help find prior art in code you know
  - Various prior art bounty programs
    - Example: Unified Patents for Rotschild v GNOME dispute

What Can I, A Non-Lawyer Do?

- Get political!
  - “Inventors Rights Act of 2019”
    - Designed to undo perceived anti-patent developments over the past 10 years
      - Patent holder must consent to any Patent Office challenge to their patent
      - Patent holder has right to pick patent-friendly court to sue in
      - Upon finding infringement
        - Injunction stopping infringement is presumed
        - Higher monetary awards available

What Can I, A Non-Lawyer Do?

- Be careful around standards
  - Many standards are royalty-bearing
  - Some standards participants are arguing use of certain free software licenses (BSD) allow them to reserve right to charge patent royalties against code they wrote and contributed

Questions?