OPEN SOURCE & PROPRIETARY SOFTWARE

Pamela Samuelson,

IS 296A Open Source Class,

October 10, 2005

IPR BACKGROUND

- Both open source and proprietary software rely upon the existence of intellectual property rights (IPRs), albeit in different ways
- Some open source business models depend on IPRs more than others
- To make informed judgments about the roles of open source and proprietary software in the economy, it helps to have some background understanding of IPRs
  - Also relevant to the conflict or coexistence debate

BRIEF HISTORY OF SW IPRs

- Phase 1 (to late ’70s):
  - software was often bundled with hardware
  - some custom software was commissioned by firms that needed it; K determined rights
  - in research settings, software was made freely available, adaptable, source code open
  - IPR status of software unclear
    - ?s about copyrightability
    - ?s about patentability

MISMATCH OF SOFTWARE & ©

- Copyright protection is available for original works of authorship fixed in a tangible medium of expression, but not for functional designs such as machines or machine processes
- Mid-1960’s: US Copyright Office decided to allow programs to be registered but did so under “rule of doubt”
  - Programs in source code were original texts, but CO recognized that machine-readable programs were machine processes (which copyright doesn’t protect)
  - Object code doesn’t convey meaning to humans

MORE ON SOFTWARE ©

- WIPO recommended "sui generis" (of its own kind") form of protection for programs in 1970s
- Japan was considering sui generis too
- CONTU Commission Report in 1979 recommended copyright protection for computer programs; Congress passed bill that implicitly accepted this recommendation
  - *CONTU Revisited* in 1984: sui generis, not ©
  - Manifesto article in 1994: why sui generis better
- International deliberations intense till 1994
  - TRIPS Agreement makes copyright for programs an international norm (although unclear as to scope)

SOFTWARE PATENT ?s

- Mid-1960's PTO considered software patentability
  - Influenced by Presidential Commission that saw no need for patents for software because industry already had ©, trade secret, & licensing
  - "Mental process," "printed matter," and "business method" limits invoked
  - © for "writings" and patent for "machines" (exclusivity theory)
- Gottschalk v. Benson (SCT 1972): algorithm for transforming binary coded decimals to pure binary form is unpatentable subject matter
- Parker v. Flook (SCT 1978): program for updating alarm limits for catalytic conversion not patentable
- Diamond v. Diehr (1981): 5-4 decision allowing patent on rubber curing process utilizing program as element

SOFTWARE IPRs IN 1980's

- Some questions still existed about © and patents for software, so licensing most common form of protection in early 1980's
- But then the mass market began to develop
  - Object code distribution only
  - Rely on © to protect code vs. duplication (but most of software internals considered trade secrets)
  - Use of "shrinkwrap" licenses (printed form purporting to grant a license conditioned on various terms, including clauses prohibiting IPRs, modifying code) of questionable enforceability
  - This was the "proprietary" software strategy

TURNING POINTS IN ©

- Whelan v. Jaslow (1986): © protects "structure, sequence & organization" (SSO) of programs and "look and feel"
  - Series of cases protected program functionality
- Computer Associates v. Altai (1992): Whelan technologically inaccurate; © for programs is "thin" because can't protect functional design elements, including interfaces
- After *Altai* became the accepted rule, proprietary software developers turned more to patents
  - CAFC increasingly receptive to software patents

SOFTWARE IPRs TODAY

- Virtually all firms rely on ©, regardless of whether open source or proprietary
- Virtually all firms rely on licenses
  - Some lingering questions on enforceability, but ?s now mostly about certain terms (e.g., anti-RE clauses)
  - Ability to use software depends on acceptance of license terms
- Proprietary firms still distribute object code only and often restrict reverse engineering & modifications; claim internals as trade secrets
- But proprietary firms obtain patents to hold in portfolio; major firms cross-license
- Open source developers may also distribute proprietary complements; struggling to deal with patents

STALLMAN'S REVOLT

- In the context of the emerging "proprietary" model, Stallman formulated alternative model for software distribution which he called "copyleft"
- Recognized that pure public domain play (no IPRs at all) would not achieve his objectives
  - It would allow proprietary software developers to make proprietary derivatives of his code
- To ensure this couldn't happen, his GPL invokes © as a form of protection for the code he develops
  - GPL license is conditioned upon release of source, ability to modify & redistribute code, derivatives bound to open terms

"FREE" vs. PROPRIETARY SOFTWARE

- Biggest ideological clash is between the Free Software Foundation's GPL and proprietary model epitomized by MS
  - Eben Moglen is General Counsel to FSF
- SCO v. IBM: principal legal battleground now
  - MS funding SCO to challenge Linux
  - Some fear that software patents will undermine F/OSS
- Some specific concerns:
  - To which derivatives does the GPL apply?
    - Does GPL apply if GPL code "touches" other code?
  - How to deal with royalty free patent provision of GPL?

"FREE" vs. OPEN SOURCE SOFTWARE

- Almost as big an ideological rift between "free" and open source software
- Advocates of "free" use GPL (although so do some open source developers)
- "Free" software is actually more restricted than open source software in terms of ability to make proprietary derivatives (among other things)
- Bitter feelings in the two camps

IPRs & BUSINESS MODELS

- To what extent do free or open source business models depend on IPRs?
  - Which ones depend on IPRs more?
  - Which ones depend on IPRs less?
- Why are IPRs important in some business models, but not in others? Which IPRs are most important, which less so?
- Do you agree with the observation that if you have a good business model, you don't need IPRs?
- If you had to choose between a "good" business model and IPRs, which would you choose?

MOGLEN'S ESSAY

- What does he mean by "IP droid" and "econodwarf"? What is he trying to say about them and their perspective?
  - In what respects is he right or wrong?
- Does he believe that no digital information can be "property"? Why? If not, why does it seem as though he does?
  - Long #s as an example of identicality
- Is there an inconsistency between his skepticism about software as property and the fact that the GPL that invokes © as basis of license?

COEXISTENCE OR CONFLICT

- What factors suggest that F/OSS can coexist in the market with proprietary software?
- What factors suggest that F/OSS and proprietary software are in irreconcilable conflict?
- Are there some markets where the proprietary model may work better and some where the F/OSS may work better?
- Is F/OSS more likely to be sustainable over time or proprietary software? Why?

MUNDIE'S TALK

- How is MS's "shared source philosophy" similar to and different from F/OSS?
- Mundie says F/OSS has downsides:
  - Unhealthy forking of code base
  - Weaker interoperability
  - Weaker product stability
  - Hindrance in planning for future
  - Security risks
- Do you agree? If so, why? If not, why not?
- What is Mundie's main objection to the GPL?