Discussion of:
The Law and Economics of Reverse Engineering

Xavier Vives

INSEAD

xavier.vives@insead.edu

http://faculty.insead.edu/vives

Summary

- Reverse engineering is the process of extracting know-how/knowledge from a human-made artifact
- Traditional manufacturing:
  - _ RE to make directly competing stand-alone product
  - _ Legal rule: RE is OK makes sense because RE is costly/time consuming
    - (if RE too cheap/easy, like with plug-molding of boat hulls, it should be restricted)

Information-based industries

- Rules restricting RE adopted or proposed:
  - _ Digital content is in the surface of the product
  - _ Technical protections raise cost of RE
  - _ Examples:
    - Semiconductor chip (SCPA, 1984)
    - Software industry: can decompile program code for interoperability reasons
    - Technically protected digital content
- Challenge:
  - _ Design rules to balance incentives to innovate of incumbent and entrants
  - _ Goal of intellectual property law: protect incentives to innovate

RE of software and the law

- Software distributed in object code form
- RE permits obtaining approximation to original source code
- From this information can develop interoperable program

  (very difficult to develop competing non-identical program)
- Questions:

  _ Do copies of programs made in the decompilation process infringe copyright/trade secrecy law?

  _ Can contractual restrictions in software licenses prevent RE?

Legal debate

- Intellectual property law: can decompile & disassembly program code, particularly for interoperability reasons
    _ US: for ” legitimate” purposes (Sega v. Accolade, 1992, Sony v. Connectix, 2000)

    _ European Directive (1991): for interoperability reasons
- Enforceability of contractual restrictions is contentious:
    _ Conflicting US caselaw

    _ EU Directive: anti-decompilation clauses in software contracts null and void
- Samuelson and Scotchmer:
    _ RE for interoperability should be allowed

    (on balance more beneficial than harmful effects)

The Economics of RE in the software industry

- System: platform (A) + applications(B) with interface to achieve interoperability
- Application Programming Interfaces (APIs):

  To make a program interoperate with a platform need precie details about how platform sends and receive information
- Strategy: Open or closed interface?
  - IBM, Apple
  - MS in OS: de facto standard with ” embrace and extend ” (integrating applications in Windows, bundling, control of APIs)
  - Game systems: serial monopolies

RE in the software industry

- RE in software industry involves entry at applications level rather than development of competing platform
- RE turns closed interface into open interface at a cost
- Erodes commitment of incumbent to closed system/tying/technical bundling
- Can think of degree of RE has choosing a point between closed and open systems

Tying and bundling

- Bundling:
  - Pure (credible with technical integration)
  - Mixed: bundle offered at a discount from components
- Private incentives: bundling as
  - Generating efficiencies
  - Accommodating strategy
    - Facilitating practice
    - Price discrimination
  - Exclusionary strategy
    - Vertical foreclosure
    - Leveraging market power

Tying: welfare analysis

- Short-run:
  - Decrease in prices: +
  - Decrease in variety (because mix and match not possible): typically _
  - Price discrimination: + or _
- Dynamic
  - Efficiencies of product integration for consumers, lowering costs: +
  - Exclusion of rivals (via pricing and/or innovation): typically _
  - Decrease (increase) innov. of rivals (tying firm): + or _
- Rule of thumb:
  - Efficiencies presumed if there is no exclusion of rivals

Tying and innovation

- Tying decreases (increases) innovation of rivals (tying firm)
  - Tying makes succesful entry prospects in complementary components markets A and B more uncertain and discourages investment by entrants because they have to succeed in both markets

    (Carlton-Waldman (2000), Choi-Stefanidis (2001))
  - Incumbent when innovating in B (applications) internalizes profit generated for segment A (platform)

    (Choi (1996, 2000), Farrell and Katz (2000))
- Welfare analysis ambiguous: what matters is aggregate incentive for R&D

RE and innovation

- RE will
  - _ increase rivals‘ R&D in platform A and applications B (easier to enter)
  - _ decrease incentives of incumbent in A and B
- Suppose closed interface yields too little aggregate R&D incentive in B and too much in A
- Can RE fine-tune incentives?
- Strike a balance between encouraging entrants’ R&D in B without killing incentive of incumbent in A

Prices

- Systems:
  - _ Closed interface (incompatible and integrated systems)
  - _ Open interfaces (compatible and unintegrated)
- Prices are lower with closed interface, because of ” Cournot internalization effect ” of bundling, but typically welfare also, because of no mix and match with heterogneous preferences

  (Nalebuff 2000, Chiovenau (2002))
- Entry deterrence/exclusion
  - _ If incumbent bundles rivals have no incentive to bundle with inelastic demand (Nalebuff (2000)) but they do with elastic demand (Chiovenau (2002))

(R) Social Calculus of Reverse Engineering of Software for Purposes of Interoperability

Social Welfare Criterion	**RE legal**
ncentives to develop platform	lower for incumbent Aggregate? higher for entrants
ncentives to develop applications	lower for incumbent Aggregate? higher for entrants
ystem Price	
Short run	higher
Long run (tipping)	lower
uplicated costs	lower ?

Evaluation

- Samuelson and Scotchmer:
  - Rule (can decompile & disassembly program code for interoperability reasons) is economically sound because it promotes development of a wider range of software
- Questions:
  - Does it strike the right balance between encouraging entrants' R&D without killing incentive of incumbent?
  - Does it make exclusionary strategies more difficult?
- Answer: probably yes as long as it is fine tuned appropriately and put in the context of the other policy levers