Open Source Software, and Standards, and IP in One Lesson

November, 2006
Beijing, People's Republic of China

Stephen R. Walli

What this talk is about

- This talk looks at the economic relationships between free and open source software (FOSS), standards, and intellectual property
- There isn't any mystery to the economics around FOSS -- the rules haven't changed, but the yardstick might be different

Always question:

"And that would be different from other proprietary or closed software how?"

Free and Open Source Software

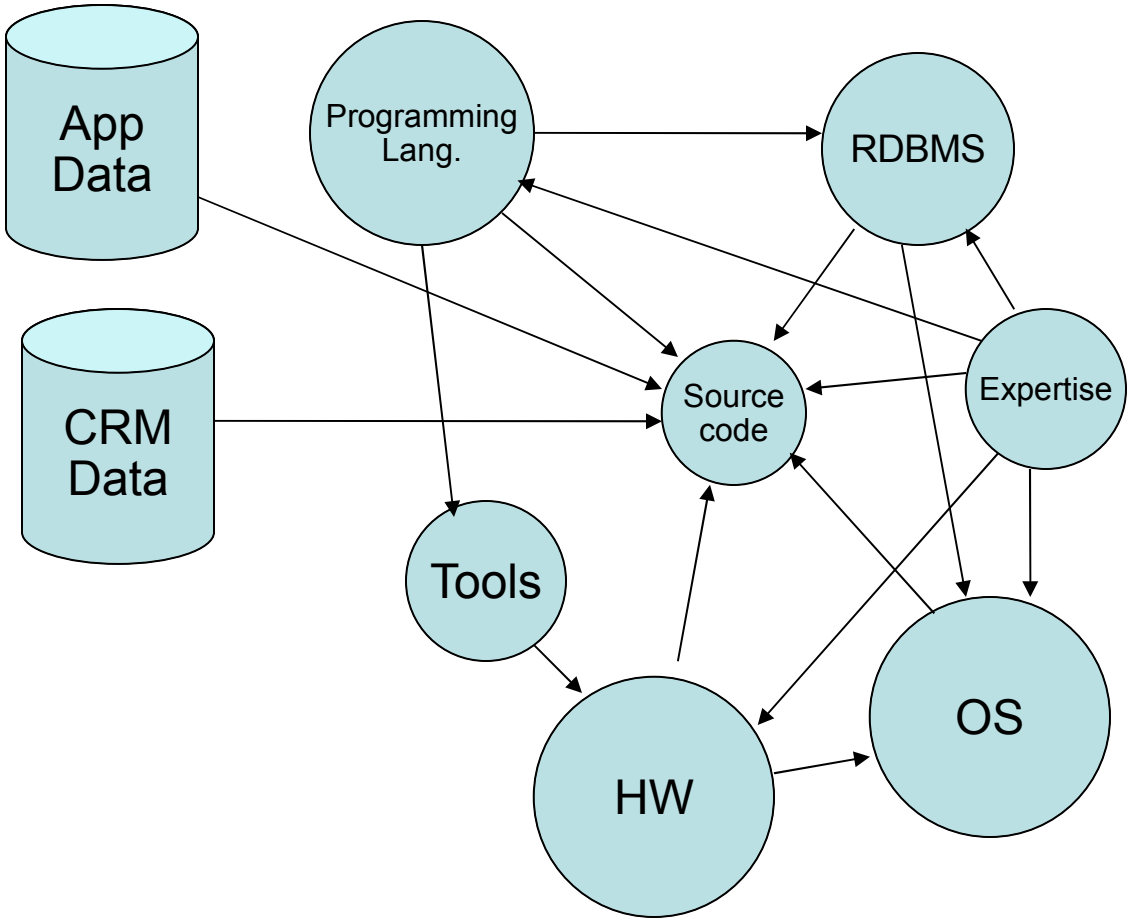
Source code for the software is always available, and the user of the software is licensed (always) to change and redistribute the software without fee, penalty or asking permission

Things to Remember about Open Source

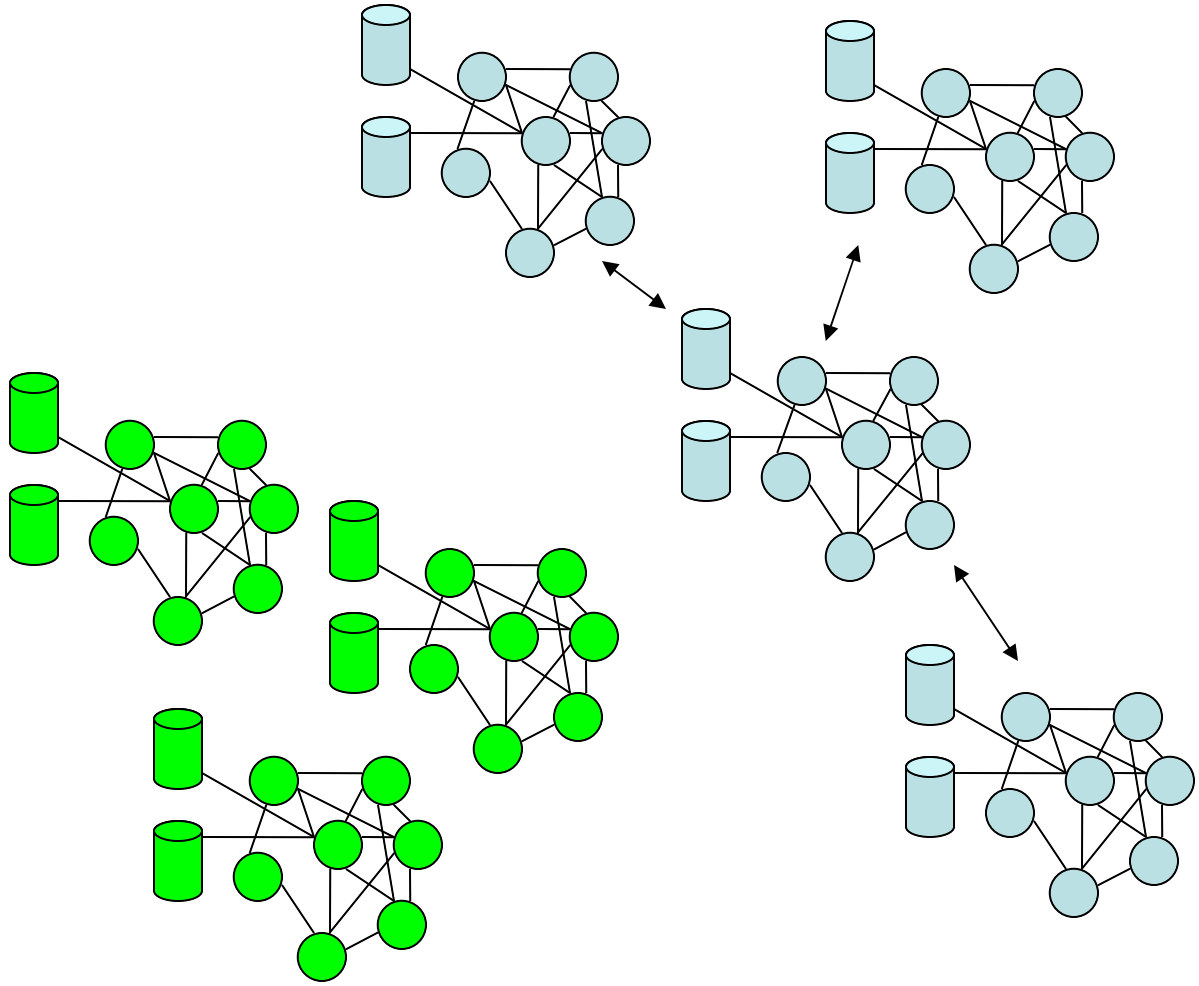
- It's just software
- It's just economics
- It's just business
- It's just software licensing

- There's nothing inherently "new" about developing good software, communities, collaboration, or software licensing

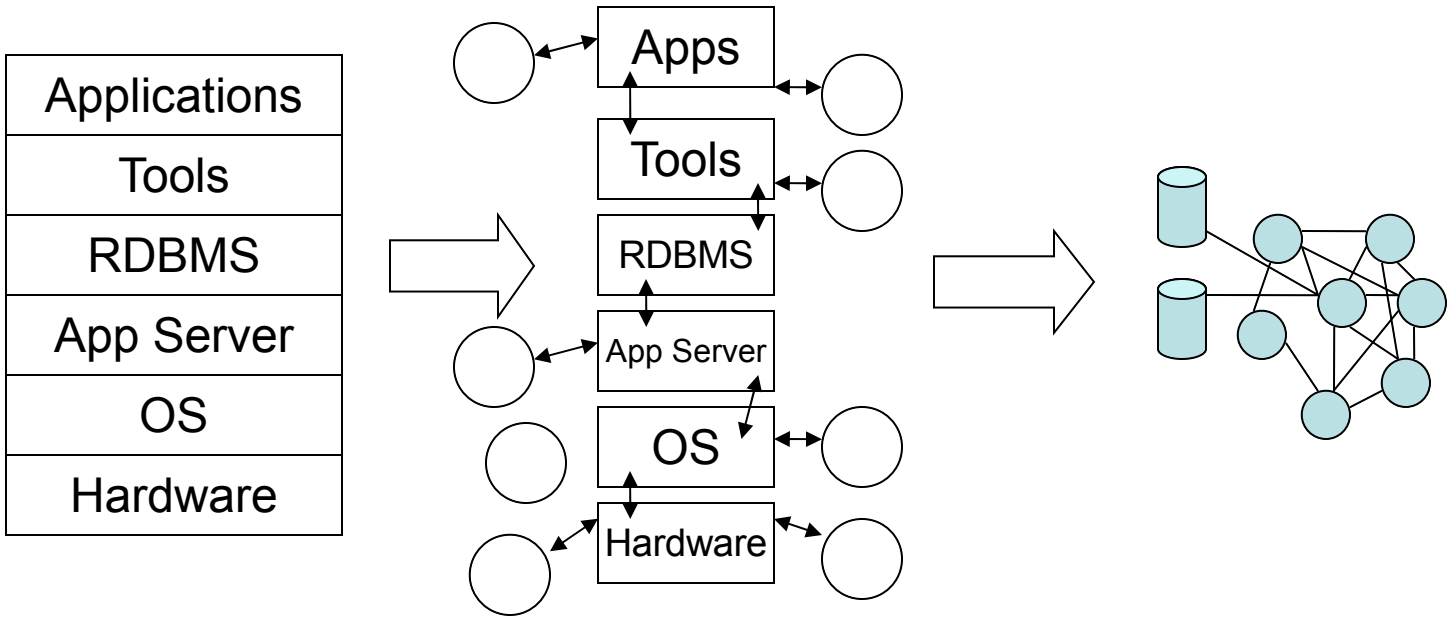
The application is actually a network …



… and the network isn't "simple"



It’s not a stack – it’s a network.



i.e. the “stack” is a view through the network.

It's Just Software

- Good software is developed by good software developers regardless of licensing strategy
- FOSS projects start when a developer writes software and is willing to share it out under a FOSS license
- FOSS projects are not products – they are interesting buckets of technology
- FOSS architecture tends to reflect a UNIX heritage of loosely couple components with stable well defined interfaces

It's Just Economics

- People value their skill sets differently in different contexts (think of all the ways a technical writer can use their writing skills throughout the day)
- No one is "working for free" in a real economic sense
- You get more than you give
- If you want to influence you have to participate and contribute
- Companies gain the same benefit as individuals

It's Still Just Economics

- *"A community is just a group of people that share a common interest -- they don't have to like each other." -- Bob Young, former CEO, Red Hat*

- A project is an interesting bucket of technology with users and contributors, i.e. the project's community

Projects to Products

- *"The early community is willing to trade time to save money; the late community is willing to trade money to save time. My customer is in the late community." -- Marten Mickos, CEO, MySQL AB*

- A product is packaged, installable, tested, documented, supported, and maintained for customers

- Companies build products as part of their value proposition to their customer; another way to say this is customers buy products (solutions), not software

Engineering Economics

- Two ratios are inescapable in programming:
  - The average number of lines-of-code a developer writes per day (~10-20)
  - The average number of bugs per thousand lines of code (1:1000 in well run projects)
- So it all boils down to writing less code
- Good FOSS projects are the ultimate reuse strategy -- good engineering practice is to design to the scarcest resource -- time!
- It's okay to do custom development again

Business Economics

- Two other apparently inescapable ratios:
  - The maintenance cost over an application's life time (~75%)
  - The number of applications that never get deployed (~60%)

- This is why packaged software took off in the 1990s: the economics worked in the customer's favour

- This is why packaged software is beginning to fail in certain vertical markets

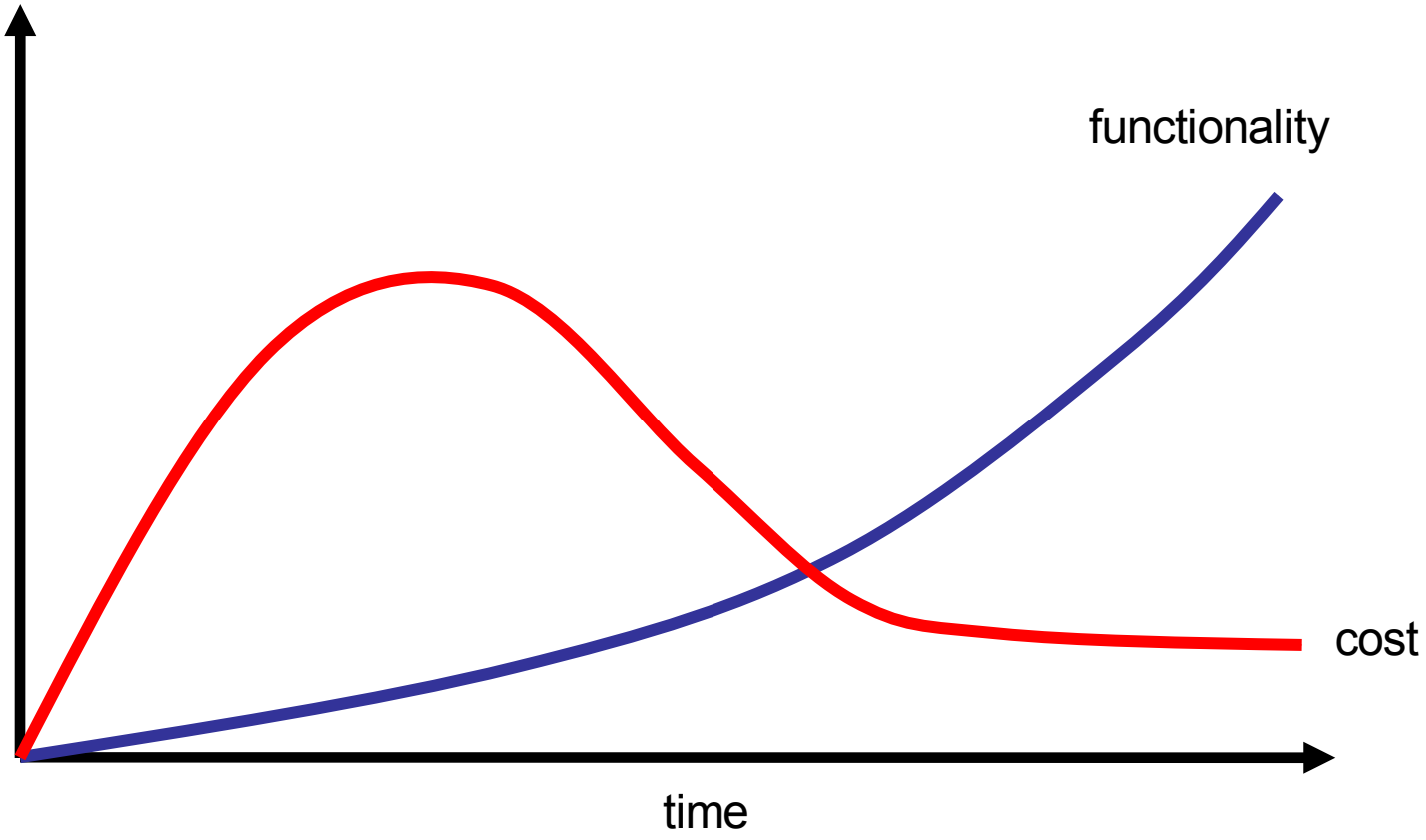
“Using” Open Source Software

- It’s “free” as in beer
- It’s “free” as in freedom
- Community support works
- Self support works
- You can buy professional support from a wealth of players big and small, broad and specialized
- You can rapidly prototype solutions, and they’re “cheap” to throw away, and cheap to deploy

“Buying” Open Source Software

- Buy it just like any other software
- The vendor has profitable business based on different margins
- While you may not “buy” the software -- you do buy the product
- Buy (competitive) support and maintenance with your commercial open source product

“Making” Open Source Software



Share the costs of value creation, maintenance, and support

“Whole Solution” Business Tools (for Vendors)

- Traditional buy-versus-build strategies through the vendor’s own brand, regardless of whether the complement products are offered as add-ons or bundled directly with the core revenue stream for “free”
- Develop a rich ecosystem of add-ons by encouraging developer and partner networks to provide a bigger whole solution
- Publish proprietary specifications enabling more partners to develop stable businesses in the complement spaces.
- Developer tools that help add complements to the ecosystem.
- Certification programs around the core technology creating service professionals to help customers complete and support their solution
- Develop a consulting services arm for part of the solution
- Develop training programs and train-the-trainer certifications

FOSS Business Tools (for Vendors)

- To buy-vs-build you add “borrow” and “share”
- Companies that participate in communities
  - Can rapidly develop complements to core offerings in their solution network (without necessarily building complete products)
  - Can amortize dev/support/maint costs of software components across customers/partners/competitors
    - IBM and Apache and Websphere
    - The vendors in the OSDL today are no different than the vendors in the OSF 20 years ago sharing the development costs of OSF/Motif and OSF/1
  - Get to interact directly with like-minded customer prospects in community, influencing customer/partner developers
- Companies that participate deeply in communities better influence those communities (e.g. participate, hire, or acquire)
- Companies/Foundations that legally own the property control the property

FOSS Business Tools (for Vendors) - 2

- Companies can use FOSS projects to reduce the cost of sales by allowing users to try easily and pre-qualify themselves as customers
- FOSS projects are an interesting publication strategy against competitors from an IP strategy perspective
- New companies with lower margin business models (compared to the incumbent) can use FOSS components to rapidly develop products that either serve new markets or serve the bottom end of an over-served market and then evolve over to or up into an incumbent's market space

It's Just Software Licensing

- Getting sued over FOSS
- Intellectual property and open source
- Patents and Indemnifications and Insurance
- FOSS licensing depends upon strong copyright laws
- “Dual” licensing is an attribute of IP law: you can license your property to as many people as many times and as many ways as you choose
- Collected works also have licenses
- It's all just software licensing – there is nothing particular (or peculiar) to open source here

First, the disclaimer

- **I am not a lawyer:** I am a business manager and development manager that has worked around free and open source software for 15+ years
- **This presentation is not "legal advice":** It is the business experience gained from working with free and open source software, proprietary software, and a mix of both in the enterprise, and various software companies including Microsoft
- When in doubt, talk with your lawyers

If you only remember ONE thing in this part of the talk …

It's all just software and software licensing:

- There is no appreciable difference between free and open source software and any other software when it comes to licensing and legal risk
- Indeed free and open source software is often less risky
- The transparency may at first make things look more complex

The Absolute Basics

- Free and open source software (FOSS) licenses depend upon strong intellectual property (IP) law: The idea that FOSS is IP hostile is a Myth
- Intellectual Property is a set of legal "tools" that one strategically applies to intellectual assets: Not every asset needs to be considered "intellectual property"
- Types of IP:
  - Copyright = How you protect the expression of an idea
  - Patent = How you publish an idea in a legally protected way so others may not build it
  - Trademark = How you protect the way you identify an asset
  - Trade Secret = How you legally protect an idea as a secret
- Companies use a combination of these tools to protect their product in the marketplace
- Software is considered to be protected by copyright law
- In the U.S. one can create a patent for an idea expressed in software

Intellectual Property versus Innovation

- *“People don't want to buy a quarter-inch drill. They want a quarter-inch hole!” – Theodore Levitt, HBS*
- Innovation is a supply side activity
- Customers don’t care about IP – they don’t buy “innovation” – they buy solutions to problems

IP Strategy, Tools, and Business (for Vendors)

- Patents, copyrights, trademarks and trade secrets are legal tools to turn intellectual assets into legal property to “protect” them
- As some legal property tools are more costly to manage than others, one can apply an intellectual property strategy across ones intellectual assets best once one looks at core vs. complement in one’s solution network
- IP is a vendor-to-vendor negotiation tool regardless of whether the vendors are partners or competitors
- Publication is an inexpensive tactic to prevent competitors from patenting in a space: possibly even aggressively salting the fields around them while best serving customers

Getting Sued over Open Source Software

- The SCO Group is the discussion point around which all discussion revolves.
- The Canopy Group has a historical business model based on technology acquisition and litigation. They were successful against Microsoft and CA.
- SCO Group began litigation against Autozone and Daimler-Chrysler as a pressure tactic against the IBM suit.
- Other vendors with threatened business models have used this as a rallying point for discussions around indemnification and dressed it up in the rhetoric of property ownership.

The Legal Risks of Proprietary Software

- Companies stand a greater risk of being sued over license counting issues from their commercial software vendors.
- Some proprietary software vendors use the threat of BSA auditing and litigation as a stick.
- Open source licensing **reduces risk** in these areas.

License Management and Compliance

- License management and Compliance: The concerns raised over tainting your code, having to publish your own secrets, and infringing property and mostly vendor centric propaganda.
- The code taint problem isn't directly related to open source, but to software development in general.
- Open source isn't the problem here: think about all the other sources of "taint" a company encounters from portals, third party code, and products.
- Publication risk is only triggered on distribution and even then, you can withdraw and correct the code usage.
- Software product companies have different risk profiles to enterprises with respect to infringement.
- Education vs. Tools: Developers, Managers, and Lawyers

Patent Litigation in Reality

- Patents are negotiation tools between vendors: not vendors and customers
- A couple of thought experiments for consumers:
  - Buying the more patented product?
  - The opportunity to buy something twice? (Or thrice?)

#1 Did you buy the Honda over the Toyota because it had more patents?

#2 If Toyota then tried to license their patents directly to you: would you?

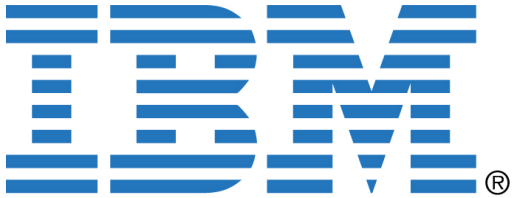
Indemnifications and Open Source

It all started with the SCO Group suit ….

Offering Indemnifications

vs.

Insists Indemnifications Unnecessary



Insurance

- Insure assets (not liabilities) based on real value and measurable risk
  - One may increase life insurance over time as earning power increases
  - One may decrease the auto insurance over time as the value depreciates
- A Lloyds of London underwriter in conjunction with Open Source Risk Management has recently created an insurance offering for software development companies based on license compliance and the GPL
- Pitched as business interruption insurance in the event that you need to expose your source code or withdraw your product due to GPL taint, and involves compliance inspections by OSRM staff (surprise!)
- One needs to evaluate such insurance against the industry loss data and the opportunity costs of other process improvement or risk management costs

"All complex ecosystems have parasites."
Cory Doctorow

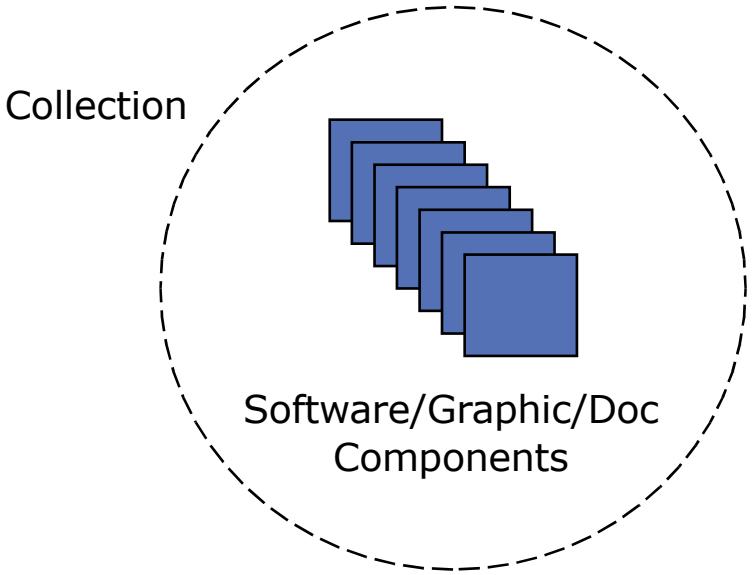
“Dual” Licenses

- It is an attribute of IP law that the property holder can license their property as many times and in as many ways to as many people as they choose
- MySQL AB licenses MySQL (the database engine) under both the GPL if the licensee is using or developing free software, or under a MySQL OEM license if the licensee wishes different terms
- This is often referred to as “Dual Licensing” …
- … but when you buy a copy of Windows XP in the office supply store it is covered by a Microsoft End User License Agreement (EULA) which is very different from the Enterprise Agreement signed by your corporation for your work machine …
- … so “Dual Licensing” isn’t really an free and open source software issue

Collected Works

- One can also copyright a “collection” of works where each work has its own copyright (e.g. an encyclopedia)
- Software products and distributions that contain free and open source components often carry their own licenses:
  - Red Hat Advanced Server has a license
  - Fedora Linux (also produced by Red Hat) has a different license
  - Novell OpenSuSE Linux has a different license again
  - Microsoft Windows XP is covered in the same way

Bringing it all together with an example



Collection:

- Protected by collected work copyright
- Protected by trademark
- A license governs the use of the product (collected work)

Components:

- Every one is protected by copyright
- Each one may be protected by trade secret
- Patents may be applied to aspects or combinations of components
- Each one might have a different 3rd party license

The Free/Open Source Difference:

- The component licenses are generally free and open source licenses (i.e. most source code is accessible and usable)
- There is probably little trade secret protection (i.e. most source code is visible)
- There are probably no software patents

Community and Contributions

- **Community Roles:**
  - Users: They use the software
  - Testers: File bugs in the bug database
  - Contributors: Contribute bug fixes, enhancements, documentation, answer questions, etc.
  - Committers: Developers with check-in access to the software repository
  - Leadership: The developers that keep it all working
- **People can have more than one role at any one time**
- **Contributors generally need to assign or license their contributions to the project in large well run projects.**
  - Originality
  - Right to assign
  - To the best of the creators knowledge, nothing infringes others rights
- **The project license is an outbound document that says how the software can be used**
- **The assignment document is an inbound document to provide clear provenance for the software from a legal perspective**

Standards

- A standard is a specification that has been put through a consensus process by a collection of interested parties
- Interested parties might be a formal de jure process of by governments, an industry or trade organization with broad interests, or a consortium of narrow focus
- Standards exist to encourage and enable multiple implementations, regardless of the developing forum, and they benefit the customer
- A successful standard (economically) has many implementations; if there is only one implementation it is a failure (or worse a vendor specification)
- Patents and standards are opposite ends of the economic spectrum (Patents protect a single implementation of an idea.)

Not Standards

- A vendor specification (regardless of whether it has been through a standards process or not) benefits the vendor by encouraging complements
- De facto technologies are NOT standards -- It refers to a single implementation with little consensus
- Corporate standards are ambiguous entities that may be selecting de facto products to reduce procurement problems, or may be pointing to real standards to encourage procurement choice

Standards and Existing Practice

- Good standards happen when a technology space matures to the point we can now see good abstractions
- If you try to standardize too early, you will fail against well engineered products
- Standards are based on “existing practice and experience”, e.g.
  - POSIX/UNIX versus a myriad of proprietary operating system interfaces
  - ODF versus troff, runoff, GML, then SGML, etc.
  - TCP/IP versus SNA, DECnet, Banyan, etc.

Standards and FOSS

- When an incumbent over-delivers in a market space, customers want “standards” so as to introduce competition: they signal this by complaining about price
- Competitors co-operate around an existing similar collaborative development project to form a standard to crack open the incumbent market position
- Essentially, the competitors attack the core revenue stream of the incumbent with a standard based on (but not identical to) collaborative technology they can share
- Collaborative technology == FOSS in today’s market
- This means FOSS projects often conform to standards (rapidly) and lead quickly to implementations
- The incumbent still thinks its about delivering better innovation thereby exacerbating their own problems
- FOSS does NOT replace the need for standards -- It fosters their creation

Examples of Standards and Collaboration

- The UNIX Wars were not about “UNIX”
  - DEC was the dominant mini-computer supplier and had over delivered on VAX/VMS
  - Competitors chose a technology base around which they had collaborated to create a standard
  - UNIX implementations quickly aligned to the standard
- ODF and Microsoft Office
  - Microsoft is the dominant productivity office suite and has over delivered with Microsoft Office
  - Sun, Novell, IBM chose a technology base around which they had collaborated to create a standard
  - OpenOffice.org quickly aligned to the standard, and implementations are appearing rapidly

In Summary ….

- It’s just software -- but open source collaborative development is proving to be the best re-use strategy for vendors and customers alike
- It’s just economics -- to “build” versus “buy”, we can again add “share” and “borrow”
- It’s just business -- the value discussion with your collective vendors hasn’t changed, but the items you’re buying have
- It's just software licensing -- FOSS relies on strong IP law
- FOSS doesn’t replace the need for standards, but helps to create them

Always question:

“And that would be different from other proprietary or closed software how?”

Questions?

- Email me: stephen.walli@gmail.com
- “Once more unto the Breach” -- a web log about open source, standards, and the business of software at http://stephesblog.blogs.com