Core, Complements, and Context: Perspectives on Software Business Models

April, 2007
Helsinki, Finland

Stephen R. Walli

What this talk is about

- This talk looks at technology business models and how to apply free and open source software (F/OSS) to business
- There isn't any mystery to the economics around F/OSS -- the rules haven't changed, but the yardstick might be different

Free and Open Source Software (F/OSS)

The 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

Ideas to Remember about Open Source

- My Bias:
  - It's just software
  - It's just economics
  - It's just business
  - It's just software licensing

- Business execution is hard work
- There's nothing inherently "new" about developing good software, communities, collaboration, or software licensing
- F/OSS is an excellent tool to develop businesses

MKS, Interix, Microsoft, and Optaros

- MKS and the “UNIX Tools on DOS” evolves into InterOpen (the UNIX Wars, “Open Systems”, and AT&T vs. BSDI lawsuit)
- Interix implemented a UNIX environment on Windows NT to migrate UNIX applications to Windows and have them behave with fidelity (Juggling a Microsoft Windows source code license, our own assets, and ~300 utilities and libraries covered by ~25 different “free” or “open source” licenses)
- Microsoft: “Shared Source”, ROTOR, and open source strategy (WiX)
- Optaros: Developing application solutions for customers with free and open source software – managing the community contribution

Crossing the Chasm

- A classic book by Geoff Moore written in 1991
- The adoption curve for a product starts with early adopters, then the early majority, the late majority, and the "luddites" -- there is a gap between early adopters and the early majority (the "chasm") and it's based on risk
- He defines the idea of the whole solution for the customer (i.e. your "core" and all its complements)
- Core here is the core value proposition – it is by definition customer facing
- There are a collection of strategies that companies develop to produce the "whole solution"

“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 (N.B. this is NOT a standard – as much as the vendor will claim it to be)
- 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
- Certification programs around the core technology to identify products that “work together” with the core
- Develop a consulting services arm for part of the solution
- Develop training programs and train-the-trainer certifications

F/OSS Business Tools

- 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)
    - SAP and MAXDB from MySQL
  - Can amortize dev/support/maint costs of software components across customers/partners/competitors
    - IBM and Apache and Websphere
    - What Oracle *should* do with JBoss
    - The vendors in the Linux Foundation today are no different than the vendors in the OSF 20 years ago sharing the development costs of OSF/Motif and OSF/1 as royalty free base technology
  - 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

F/OSS Business Tools - 2

- Companies can use F/OSS projects to reduce the cost of sales by allowing users to try easily and pre-qualify themselves as customers
- F/OSS 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 F/OSS 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

Dealing with Darwin

- Geoff Moore's “new” classic (2006)
- Core is defined as “any process that contributes directly to the sustainable differentiation leading to competitive advantage in target markets”
- Context is defined as “all other processes required to fulfill commitments to one or more stakeholders”
- WARNING: he's now defining core as core competency – this is DIFFERENT from core value proposition
- This core is not customer facing – it is about the competitive advantage that enables the core value proposition

Value Proposition versus Competency

- Microsoft
  - their core value proposition is an incredible business desktop appliance
  - their core competency is delivering packaged software that works out of the box across the entire hardware compatibility list crossed with the application compatibility list
- Google
  - their core value proposition is a great advertising engine
  - their core competency is the ultimate search machine
- Softway (Interix)
  - “easily migrate your UNIX apps to NT” versus “deep UNIX knowledge (kernel + standards) AND Microsoft environment subsystems”

The “Rules”

- Understand the difference between the value proposition to your customer (i.e. the business model) and what competencies get you there
- Understand your value proposition is WHY customers give you money – it is hopefully not different to your “messaging”
- NEVER share/publish your core competency (e.g. Windows versus Google or a crypto company)
- People will always pay for value

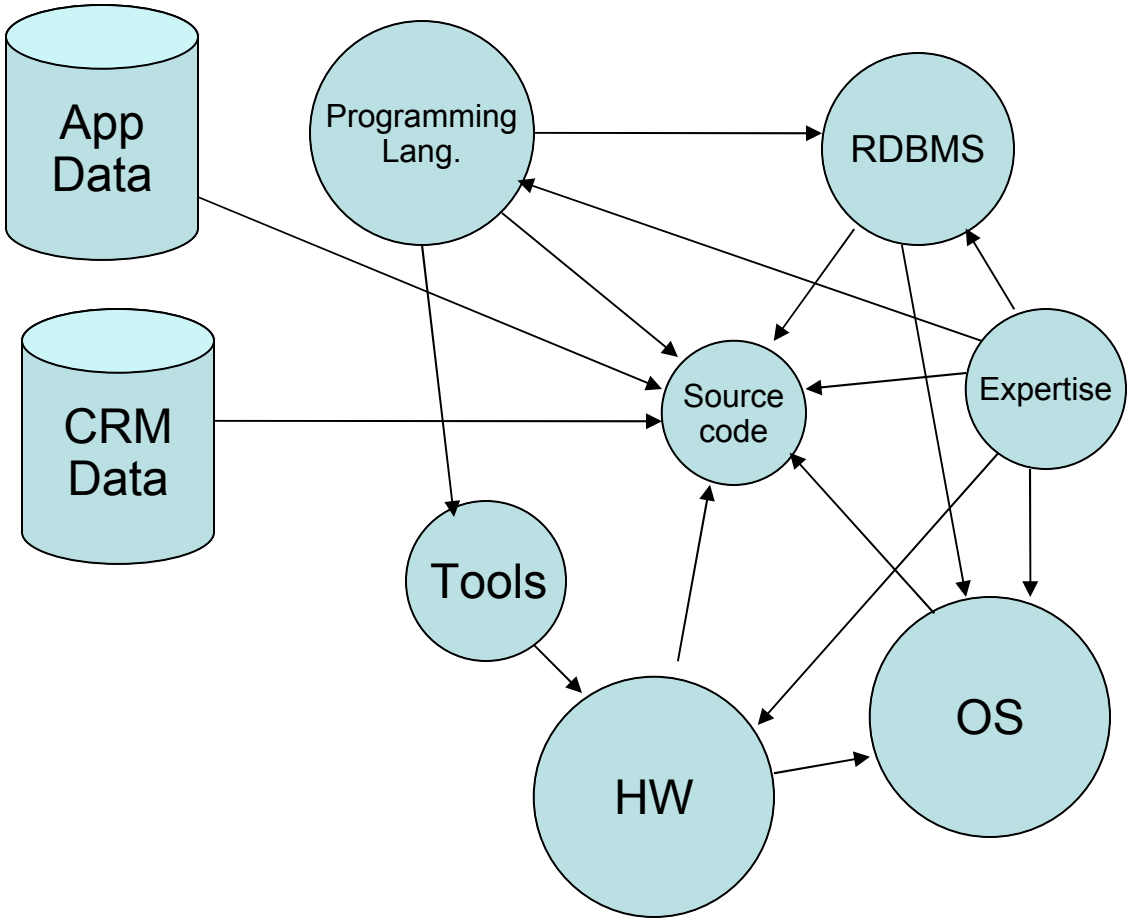
Why “Packaged Software” was GREAT!

- Packaged software took off through the 1990s because the economics served customers best
  - A vendor was in the best position to collect requirements from a broad spectrum of customers and amortize the cost of development, support, maintenance across the customer base
  - computers were relatively expensive and architectures divergent
  - the Internet wasn't the distribution mechanism it has become
- Customers got more and better software than they could possibly develop themselves

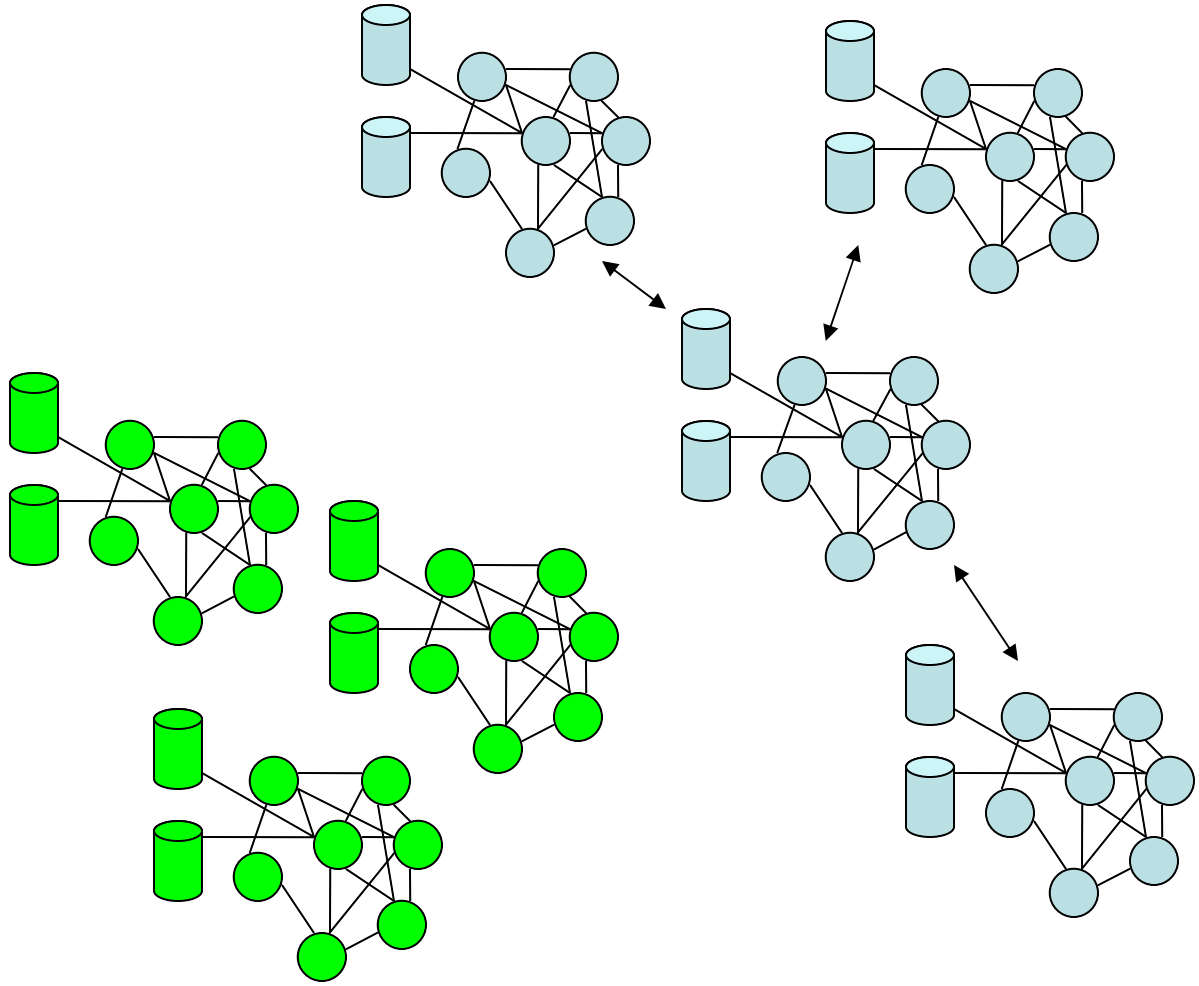
But ...

- Once a vendor over delivers on a product they become vulnerable
- Customers complain about “price” and vendors change licensing models – the account execs are compensated to pursue the high margin business – this makes things worse
- The vendor is vulnerable on three fronts:
  - standards will be encouraged by their competitors
  - new disruptive lower margin business models can eat up from the bottom
  - new different margin business models can make a product line irrelevant

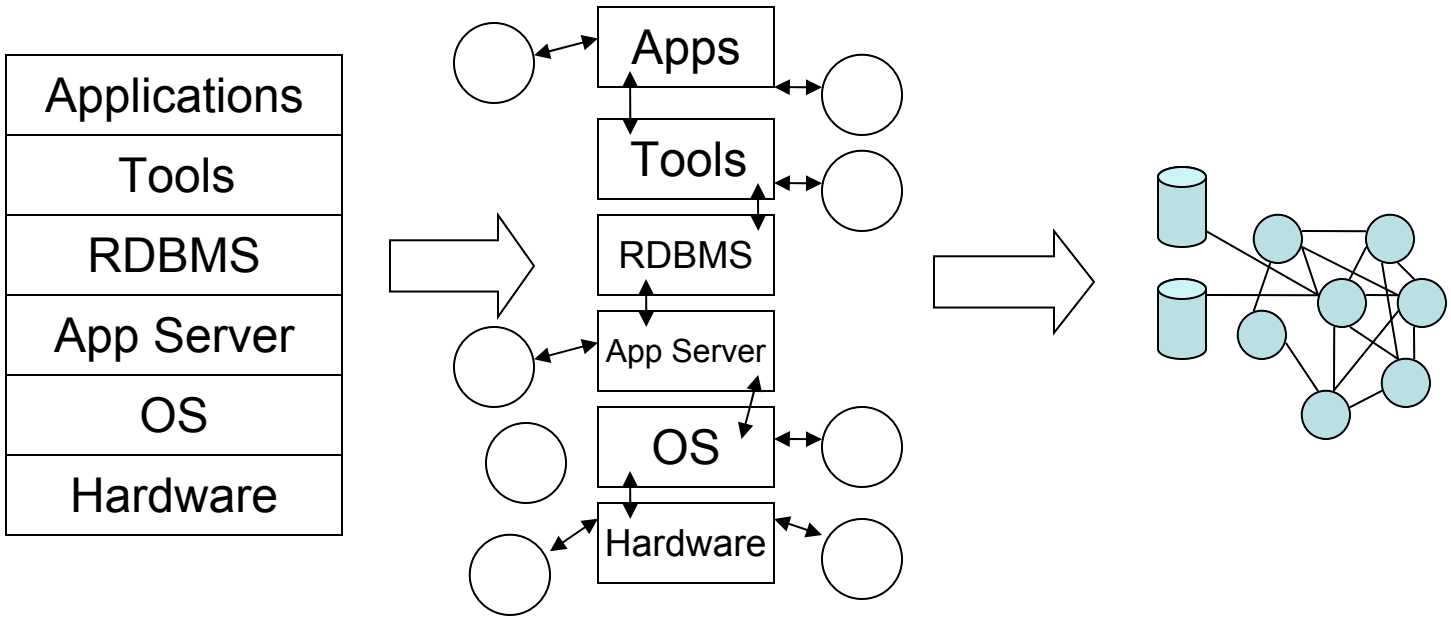
The application is actually a network …



… 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.

Projects and 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

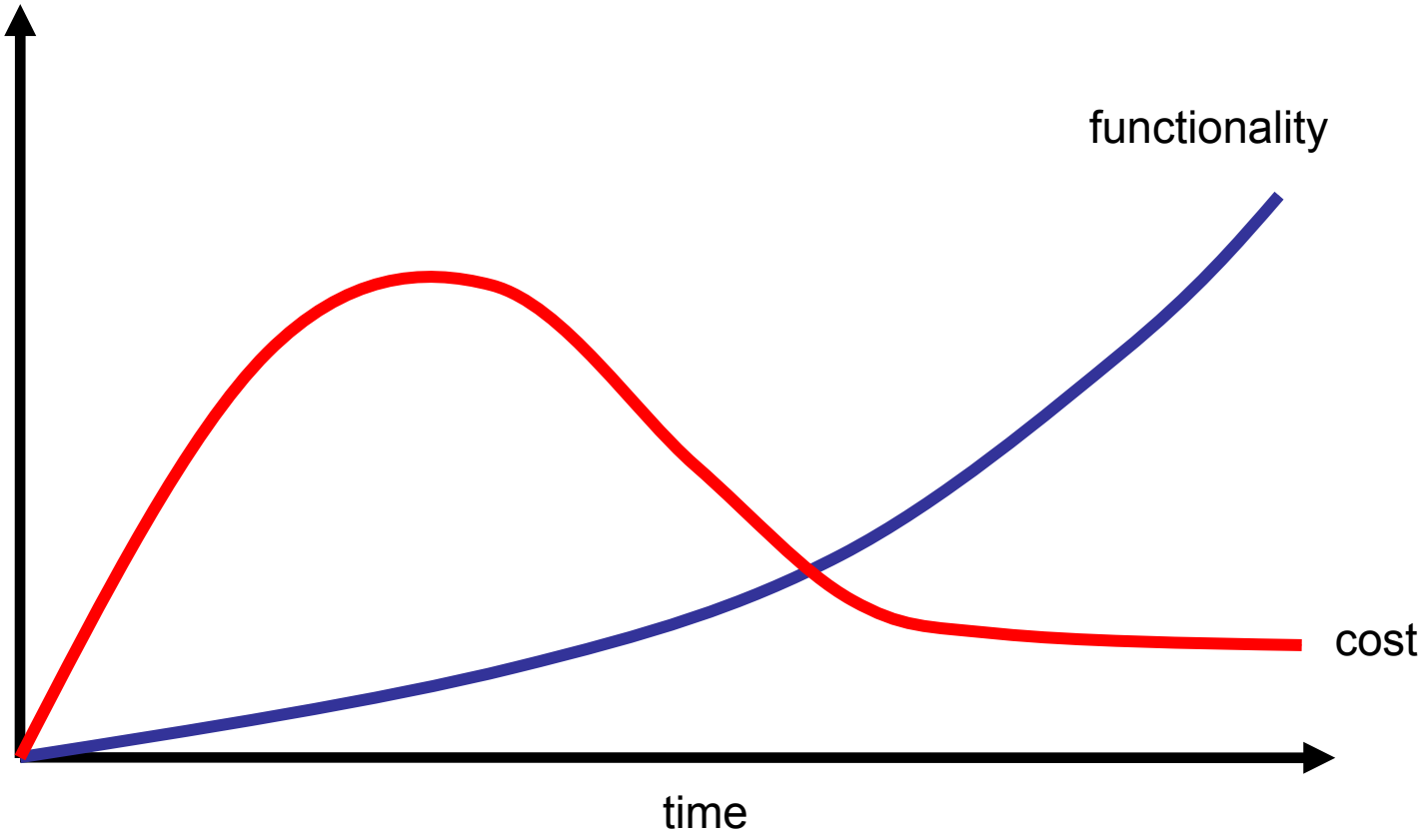
“Using” Open Source Software

- 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

It's Just Software Licensing

- 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

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

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

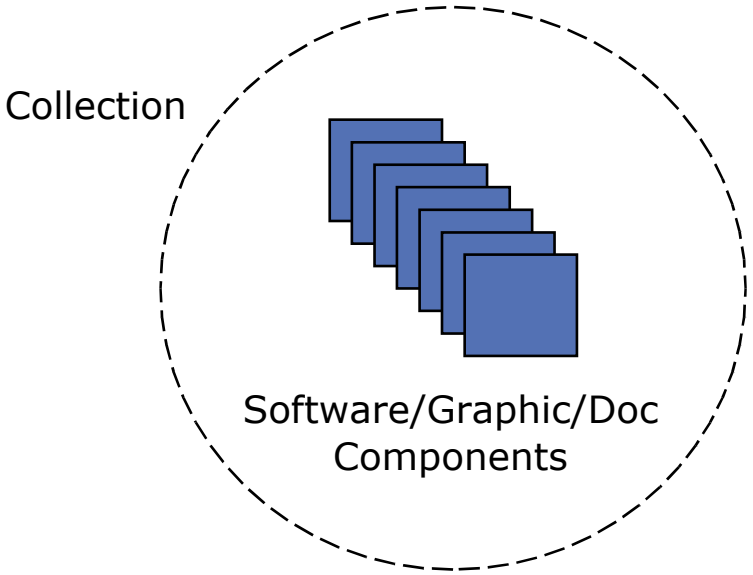
“Dual” Licenses

- 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**

In Summary ….

- Know what business you're in – understand the problem you solve for your customers and the competencies that get you there
- It's just business – and execution is hard work
- 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 software licensing -- FOSS relies on strong IP law

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