

Layer 8 (Financial) and layer 9 (Political) of the OSI protocol stack

Presentation for the Open Source Weekend¹ 2004 by Russell McOrmond².



Copyright 2004, Russell McOrmond

This work is licensed under a [Creative Commons License](#).

Abstract:

The International Organization for Standardization ISO³ Open systems interconnection (OSI)⁴ network layering model was first conceived as having 7 layers: Physical, Data Link, Network, Transport, Session, Presentation, Application. Many have suggested that there are two additional layers, namely Financial (layer 8) and Political (layer 9)⁵.

Lawrence Lessig⁶ has said that "code is law", suggesting that what we can and can not do with technology is governed not by acts of parliament but by policy embedded in computer software. We need to ask accountability and transparency questions around who sets that policy, and whether that policy is under the control of citizens or software vendors.

I would like to take these concepts and explore with the audience Information and Communications Technology (ICT) business models and public policy. Are there things that we can learn about computer mediated communications from recognizing all 9 layers that technology people would miss only looking at 7? Are there things that policy makers and private citizens are missing by thinking of computer software as a form of technology, similar to hardware, rather than as yet another form of public policy?

1 More information about the Open Source Weekend can be found at <http://www.osw.ca/> (Accessed Feb 9, 2004)

2 Full contact information for the author can be found at his work website of <http://www.flora.ca>. He is a self-employed businessperson who focuses on Free/Libre and Open Source Software (FLOSS) from a technical, business and public policy perspective. He is also the private sector co-coordinator of the GOSLING Community (Getting Open Source Logic INto Governments) <http://www.goslingcommunity.org/> (Accessed Feb 9, 2004)

3 What ISO's name means <http://www.iso.ch/iso/en/aboutiso/introduction/index.html#three> (Accessed Feb 9, 2004)

4 Open systems interconnection (OSI) <http://www.iso.ch/iso/en/CatalogueListPage.CatalogueList?ICS1=35&ICS2=100> (Accessed Feb 9, 2004)

5 Jim Gogan of the University of North Carolina includes the 9-layers on a page associated with one of his courses
7-layer model: <http://www.unc.edu/~gogan/osi.html> (Accessed Feb 9, 2004)
9-layer model: <http://www.unc.edu/~gogan/osireal.html> (Accessed Feb 9, 2004)

6 Lawrence Lessig has a website and Weblog at <http://lessig.org> (Accessed Feb 9, 2004)

What are the ways in which those of us in the "commons-based peer production"⁷ movements such as Free Software⁸, Open Source⁹, FLOSS¹⁰, Creative Commons¹¹ and others think of these questions different than those who think of knowledge as a manufactured good?

The following is a scratch-pad of ideas I am thinking of for the workshop. Please send me feedback and let me know what ideas you would like to pursue. I will be creating slides based on the headings in these notes, and will present based on that.

Disclaimers/warning

This talk is not a technical talk, and will focus on economic, political and legal issues. I Am Not A Lawyer (IANAL), This Is Not Legal Advice (TINLA), nor am I a professional politician or an economist. My background at University was primarily as a computer science and psychology student, with a little bit of Mass Communications on the side.

I am a self employed Internet consultant. While I make most of my money doing system administration and ISP hosting and co-location services, my passion is the social, political and economic aspects of FLOSS and FLOSS public policy.

The ideas I will be introducing are not "scientific fact", but ideas I have collected from others that I have thought about and wish others to think about. As being this type of conversation it is expected that you will not agree with something I have said and will want to challenge me. This is both expected and encouraged, but lets not spend too much of our time arguing about different meanings of words and other such things. My first trial balloon for some of the concepts discussed here ended up causing a long thread that amounted to differing ideas on what the word "intangible" meant¹².

With these types of talks there are also different ideas that come into peoples mind when using different phrases. If the way in which I mean something is not clear, please ask. We do not all have the same

7 The term "commons-based peer production" was first introduced by Yochai Benkler in his paper "Coase's Penguin, or Linux and the Nature of the Firm" <http://www.benkler.org/CoasesPenguin.html> (Accessed Feb 9, 2004)

8 Where used in this paper the term "Free Software" is being used as defined by the Free Software Foundation in their Free Software Definition <http://www.fsf.org/philosophy/free-sw.html> (Accessed Feb 9, 2004)

9 Where used in this paper the term "Open Source" is being used as defined by the Open Source Initiative in their Open Source Definition http://www.opensource.org/docs/definition_plain.html (Accessed Feb 9, 2004)

10 Free/Libre and Open Source Software (FLOSS) is explained in more detail at <http://www.flora.ca/floss.shtml> (Accessed Feb 9, 2004)

11 Taking inspiration from the FLOSS movement, the Creative Commons tries to encourage creators to use licensing to help build a commons. <http://creativecommons.org/> (Accessed Feb 9, 2004)

12 I posted a message titled "Canada needs to clarify: Is software a product or a service" to the discuss@canopener.ca discussion forum. The thread can be viewed at <http://lists.canopener.ca/pipermail/discuss/2004-March/date.html> (accessed March 26, 2004)

concept of what is meant even when we commonly use the phrases "Open Source", "Free Software" or even the word "software".

What is Software?

A simple dictionary definition is: The programs, routines, and symbolic languages that control the functioning of the hardware and direct its operation.¹³

Software is an intangible or incorporeal¹⁴, meaning it is neither physical nor is it easily defined. In the context of the copyright act it is called a "work of the mind". Others consider software to be a branch of mathematics as both use numbers and symbols to represent things, but this does not sufficiently differentiate software from economics and most other natural or social sciences which also use mathematics.

The nature of software is that there are a few traits that we can discuss, but few exact analogies to other things. Many different disciplines will treat it like something else, but that does not mean that software is only like the thing it is being compared to.

In thinking about what software is it is important to separate software from the mediums it is stored on or it is executed in. When software is printed on paper, burned onto a CD, or executed in a CPU, the paper, CD and CPU are all tangible. That the medium of storage and execution are tangible does not mean that software is itself tangible. It is an important to not allow the physical aspects of the tangible medium to limit our thoughts about software.

History: beginnings of the software sector

Prior to the 1950's/1960's, software was always bundled with hardware. You had a device that could do word processing, but it could not be easily reprogrammed to do other work. Hardware was obviously manufactured, distributed and sold on a per-unit basis and thus the bundle of hardware and software was marketed that way.

Some say that it was not sold at all, but just considered shared knowledge between the few practitioners of the day in much the same way as Open Source is treated today.

Some time in the 1960' s software became unbundled from hardware and two different branches of the software sector were formed. There were those who felt that software should be treated the same as

13 The American Heritage® Dictionary of the English Language, Fourth Edition , as offered at <http://dictionary.reference.com/search?q=software> (Accessed March 26, 2004)

14 Dictionary definitions are useful to look at as different people have different ideas of the meanings of these terms. Fortunately there are some on-line dictionaries that can help: <http://dictionary.reference.com/search?q=incorporeal> (Accessed March 26, 2004) <http://dictionary.reference.com/search?q=intangible> (Accessed March 26, 2004)

In any case, don' t spend too much time thinking about these specific words as you could spend years trying to get a full grasp of whether anything we can talk about is really intangible, and whether there is a difference between being touched by a hand and by the mind. You can probably get a masters just discussing "what is the mind".

hardware, with those in the "software manufacturing" movement relying almost entirely on business models and methodologies from the manufacturing sector. Software code is kept as secret as possible and the users of the software are legally disallowed from learning how the software works or modifying it to put the software under the control of the user. This subset of the software industry was very successful from the 1960' s up to today.

Another group of people noticed that software, being intangible and naturally non-rivalrous, has very different attributes than hardware. They felt that there was no need to arbitrarily limit software and software business models to those from the manufacturing sector, and think of per-unit royalty payments as simply one business model among many.

What are some other ways to think of software?

While thinking of software as a manufactured product has been popular thus far, it is obviously not the only way.

Some people think of software as simply a form of technical knowledge that describes the interfaces and logic between different aspects of the physical world. A human being typing on a keyboard will have software executing in the computer that has knowledge of keyboards which interprets this input and does something with it.

Some people look at software as a type of service. Software isn' t useful for that it "is", but for what it can do for us. People working within these methodologies would look towards the services sector and think of themselves as authoring knowledge which would automate or otherwise offer some service.

In my own work I focus on software of as a form of policy. Software is seen as the rules that govern the actions of a computer in much the same way as Roberts Rules might govern the organization of a meeting, or how an act of parliament may govern citizens. Not only does this way of looking at software change the types of business models which I believe are appropriate for different classes of software, but it also causes me to ask important questions about publicly distributed software.

Is publicly distributed software, software that can affect the governance of computers used by citizens and governments, not something that should be looked at as a form of public policy? If so, then should we not have the same citizen input, accountability, transparency, access to information and other aspects of public policy development that we expect of acts of parliament created by democratic governments?

What is "commons-based peer production"

The term "commons-based peer production" was first introduced by Yochai Benkler in his paper "Coase's Penguin, or Linux and the Nature of the Firm".¹⁵ This is an extension of some of the methodologies discussed with FLOSS software beyond just talking about software, but talking about other works of the mind such as books, music and movies. While each type of work has its own unique characteristics there are often times when similarities can be highlighted. During this discussion I will talk about peer production as being a superset of the software licensed in FLOSS licenses as well as works licensed using Creative Commons licenses.

In order to allow all creators to work as peers there is a need to grant each other various rights, and to exclude certain business models that would put certain creators as different than others. The most important feature to realize is that royalty-based business models are not going to be compatible with this development methodology. While for some types of projects it is possible to take the peer produced knowledge, bundle it with other knowledge and packaging, and resell the bundle for a royalty, this method is considered outside of the peer production process itself.

What is Peer Distribution?

Peer distribution is often associated with peer-to-peer file sharing on the Internet, but it is more than that. It can be thought of as leveraging the "and they told two friends, and they told two friends, and so on, and so on" as an economically viable method to distribute works of the mind.

Because of the informal method of communication, knowledge distributed this way needs to be royalty free as there is no mechanism to count. Any accounting mechanism imposed on such a system would make it ineffective, so business models using this distribution mechanism should be royalty-free.

Example Software Business models

There are a number of different components of a software business model. I will contrast two examples to demonstrate some of the different possibilities. These are by no means the only options available, and it is important to not close yourself off to the many alternatives available.

Typical retail "software manufacturing"

In bullet form this is one example of typical retail software manufacturing.

- Inputs may include software already authored by others which are licensed appropriately. This can include other "software manufacturing" software that is sub-licensed, or even FLOSS software that is licensed in a way compatible with this specific business.

¹⁵ Ibid at

- New software is authored, most often by employees where their contracts grant all patent, copyright and other such rights to their employers. These are often high-tech labor that interact with their employer much like the labor in a factory.
- Software is bundled, packaged (CD, boxes), and distributed to retail markets.
- While the marginal cost of software is zero, this model amortizes the fixed development costs across a large number of "copies" by charging royalties. How much is collected in royalties has no direct connection to the actual development costs, so this model has a lot of risk involved in it which needs to be managed and accounted for.
- There is a per-unit marginal cost for packaging and distribution, storage, and all the other costs associated with retailing of manufactured goods.

How this changes with peer production and peer distribution

- Inputs include software and other knowledge already authored by others and available in a compatible license. This software will be able to include copyleft¹⁶ FLOSS that was not available to the "software manufacturing" company, but will exclude the ability to sub-license other "software manufacturing" software. The most popular FLOSS license is the copyleft GNU General Public License.
- New software may be authored, but in most cases the public pool of knowledge is far closer to the desired result than where a software manufacturing firm would start. Software creators do not always have to assign their copyright to an employer, and the ability to be self employed much higher. Retaining copyright especially allows reputation to be built by the specific software creator, and serves as an important part of the resume.
- Software may be bundled with other software, but FLOSS is able to make use of peer distribution. The network effects which are so important in software work together with peer distribution to make all users into a powerful (and free) marketing force for the project.

16 The term Copyleft is sometimes confusing for people. Copyleft is software that is under copyright law, and relies on copyright law to exist. Copyleft is a specific subset of FLOSS, and the term ShareAlike is used by the Creative Commons to describe the same concept. You will often see a reverse circled-C symbol used for copyleft, since the circled-C is used to indicate copyright.

A work that is not under copyright is in what is called the public domain, and while public domain software is still considered FLOSS, it is non-Copyleft FLOSS.

The Free Software Foundation defines copyleft as ,"a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well".

<http://www.fsf.org/copyleft/copyleft.html> (Accessed March 26, 2004)

ShareAlike is described by the Creative Commons project as: "You allow others to distribute derivative works only under a license identical to the license that governs your work." <http://creativecommons.org/learn/licenses/> (Accessed March 26, 2004)

- The marginal cost of the software is zero, and the marginal price is also be zero. Business models will focus only on the fixed initial development cost of the software. Anything that can reduce the costs of new software becomes very important.
 - Speculative development is reduced. Software is produced as a response to specific customer needs, with the developer being paid for the value-add of the additional software being added to the pool of public software.
 - Business models often switch from "supply-side" analysis where a software vendor is offering solutions to customers, to "demand side" where the end user will hire a developer to create some specific feature that they need. Doc Searls, Senior Editor of Linux Journal, once wrote that "The Linux Phenomenon is hard to understand from the outside, because it' s whathappens when the demand side of the market starts supplying itself."¹⁷
 - A new concept called "resource multiplication" comes into play where other organizations will pay their own developers to add new features, and these features become available to you at no additional cost.
- Without the marginal cost of packaging and distribution it is possible for the marginal price for the customer can also be zero. This has considerable effects on the so-called "cost of ownership" as there is no longer a reason for customers to account for individual copies of software, only ensure one-time that the software (and its license) is appropriate for their organization.

The Software Paralegal

Where much of the software industry uses manufacturing methodologies, even some of those creating, distributing and supporting FLOSS software, I focus on techniques from the social sciences. Where many software creators think of themselves as software engineers, my business model is far closer to that of a lawyer. Although I do focus on software created using FLOSS methodologies which match my business model and methodologies, any marketing is focused on my experience and qualifications, not any specific software solutions. While I have more experience with certain classes of software, I am technology neutral and software project neutral, but not methodology neutral.

This is not a new concept for software. In the manual that was released with RedHat Linux 5.0¹⁸ in November 1997 it was indicated that Free Software

17 SuitWatch--May 8th <http://www.ssc.com/pipermail/suitwatch/2003q2/000044.html> (Accessed March 26, 2004)

18 It is unfortunate, but this background material was not included in later versions of their manuals. I added a reference to this to my own document "Open Systems, Free Software, and Why?". <http://www.flora.ca/open.shtml> (Accessed March 24, 2004)

RedHat Linux is one of the most popular Linux distributions. More about RedHat can be found at <http://www.redhat.com/> (Accessed March 24, 2004)

Bob Young, co-founder of RedHat, was featured in a recent article I wrote comparing his work to that of past Heritage Minister Sheila Copps <http://www.flora.ca/russell/drafts/copps-ndp.html> (Accessed March 24, 2004)

..."changes the model of software development and distribution to one much like the model our Legal system and its industry uses. If a lawyer designs an argument that wins his case in front of the supreme court his reward is not only the fees his client pays him but also the additional clients that his achievement attracts to his practice. The `` argument' he used becomes available for any other lawyer to use without restriction, and in fact becomes part of our collective legal heritage.

I have extended this thinking beyond just how my business model works. While I only offer software services, and never charge a royalty fee, I can also be hired to not only create software but do policy analysis on the social, economic, legal and human rights aspects of the use of different types of software. I have extended Lawrence Lessig' s concept of "code is law"¹⁹ concept to offer what may be seen as a software paralegal service where I analyze the policy implications of software. I have authored many submissions to government²⁰, and been hired by the Copyright Policy Branch of Heritage Canada to do presentations on Technological Protection Measures (TPM), as well as being hired by ICT Branch of Industry Canada to do a paper and a presentation on software patents²¹.

Whether I am authoring or editing a policy document, a business model analysis, a software configuration file or software itself, I treat all these things the same. The language used may be different, but how I charge my customers and how I deliver the results to them will be the same. I charge different rates for customers depending on whether or not the result can be publicly published or not, not whether what I am authoring is written in a natural language like English or a computer language.

Like all other ways of looking at software, this is just an analogy to something else. I am not a lawyer (IANAL), this is not legal advice (TINLA), and unlike what you get from a lawyer or a paralegal pretty much all software comes with no warranty at all.

Some software public policy problems

As a software paralegal I offer the following statement on my homepage²²:

Governance software that controls Information and Communications Technology (ICT), automates government policy, or electronically counts votes, should not be thought of as something that should be bought any more than politicians should be thought of something that should be bought.

19 Ibid at

20 One of my recent submissions to parliament on Copyright reform includes a table at the end that gives a list of some of these submissions. <http://www.flora.ca/copyright2003/> (Accessed March 24, 2004)

21 I was able to publicly publish the results of this work which can be seen at <http://www.flora.ca/patent2003/> (Accessed March 24, 2004)

22 Personal home-page is at <http://www.flora.ca/russell/> (Accessed March 26, 2004)

This follows into some specific public policy ideas. I believe that in a democratic society, in order to protect communications rights such as freedom of speech, we need to ensure that **"any 'hardware assist' for communications, whether it be eye-glasses, VCR' s, o personal computers, must be under the control of the citizen and not a third party."**²³ This has considerable implications for digital copyright reform, and is the primary reason I am involved in that area of policy.

When governments create policy there is a need for transparency and accountability. If governments use software to automate this policy, the same assurances of accountability and transparency are required. Since not all politicians and government policy makers understand software there is a need for software automated policy, as well as the software itself, to be made as open to citizens as possible.

The ability for citizens to vote is a critical aspect of our democracy. When the software used to count votes is kept secret from the voting public I do not consider this an example of a legitimate business model, but a case of political corruption. I worry about the possibility of governments implementing on-line voting as the Ontario Liberal government has proposed for the future²⁴. Unless software is understood as a form of policy I expect the types of problems that the new Canadian Conservative party recently saw with ballot problems during their leadership election²⁵ to be the norm and not the exception.

The Software Environmentalist

A number of my customers are NGOs that are interested in issues such as environmentalism. Software can also be thought of in this context to come to a number of interesting and practical conclusions. In bullet form I offer the following suggestions.

- separate hardware from software, and analyze tangibles and intangibles separately.
- Reduce/reuse/recycle hardware. Reduce energy consumption by purchasing energy efficient hardware when buying new, reuse and recycle old hardware by using less resource intensive software. A graphical Linux distribution focusing on KDE and Koffice can run quite nicely on old Pentium hardware that would otherwise not be used. Using AbiWord²⁶ rather than Microsoft Office on an existing Windows 9x computer is more efficient than upgrading hardware to accommodate new Microsoft Office and Microsoft Windows versions.
- Royalty-based business models create an artificial scarcity. Given the marginal cost of software is

23 Ibid

24 In materials announcing Ontario's Democratic Renewal Secretariat it included as a goal "Ensuring that Internet voting is an option" http://weblog.flora.org/article.php3?story_id=516 (Accessed March 26, 2004)

25 This was discussed in the March 23, 2004 press release for an Open Source conference in May. "Potentially Catastrophic Effects of Computer Error To Be Debated This May at University of Toronto Conference" <http://osconf.kmdi.utoronto.ca/media.html#mar23> (Accessed March 26, 2004)

According to Laschinger, at least 7,000 members "are losing the right to vote" for the leader of their party. "The computer does not recognize that two or more different people in the same province can have the same name," adds Laschinger.

26 Abiword is a lightweight Word Processor that is compatible with many popular file formats including Microsoft Word. It is available for many computing platforms including Microsoft Windows and Linux. <http://www.abisource.com/> (Accessed March 24, 2004)

zero, leveraging peer distribution and peer production can allow for efficient software and other knowledge transfer worldwide. Majority world countries would be able to reduce environmental impact by reducing their need to pay huge fees for software knowledge and export software services rather than raw materials (strip mining, clearcuts, etc)

- UN UNDP Technology Achievement Index²⁷ (TAI) includes royalty fees, patents (information process patents), electricity usage. These are all things which environmentalists should want to reduce, and should discourage governments from considering increases in these things as an "achievement".

Software patents?

"invention" means any new and useful art, process, machine, manufacture or composition of matter, or any new and useful improvement in any art, process, machine, manufacture or composition of matter;²⁸

The four tests discussed are:

- Statutory. Subsection 27(8) of the Act defines what is not patentable as "No patent shall be granted for any mere scientific principle or abstract theorem." Some believe that this excludes mathematics, and that by extension it would exclude software. Canada does honor software patents, so this is now how the patent office is interpreting things.
- Useful. Usually this relates to industrial applicability. While software may have industrial applicability, it also has non-industrial applicability. If patent law should be applied to software, should it apply to all software? Should different methodologies for the production of software be treated differently, such as a fair dealing exemption for royalty-free software like FLOSS?
- Novel. With the low cost of entry into software, and the huge amount of software being authored all over the world using many different methodologies, it is near impossible to do an adequate "prior art" test. At the moment patent offices are not using the huge FLOSS library of software for non-patent prior art searches. Even people who support software patents will state that more than 60% of the software patents granted by the United States Patent and Trademark Office would be found invalid with an adequate "prior art" test²⁹.

27 The TAI is included in the UNDP FAQ at <http://www.undp.org/hdr2001/faqs.html##8> (Accessed March 26, 2004) and uses patents and royalties as an inappropriate proxy for "creation of technology", and electricity consumption as an almost backward proxy for "diffusion of old innovations".

28 Canadian Patent Act (R.S. 1985, c. P-4) <http://laws.justice.gc.ca/en/P-4/> (Accessed March 26, 2004)

29 Gregory Aharonian <http://swpat.ffii.org/players/aharonian/> (accessed March 26, 2004) wrote on an Internet mailing list that he believes that over 60% of the issues software patents issued by the United States patent office are invalid <http://www.aful.org/www/arc/patents/2003-01/msg00030.html> (Accessed March 26, 2004). Individuals on all sides of the discussions of software patents tend to agree with the statistics that Gregory produces.

- Unobvious. Software innovation moves at such a quick pace that what may not be obvious today can become obvious shortly as the state of the art moves forward. It becomes questionable whether any idea in software can be considered unobvious within the 20 year term of a patent.

Many of those who support software patents talk about their belief in a physical nature of software, and how software "controls the forces of nature" like manufacturing in that software controls whether a CPU will direct an electron one way or another. Given the human brain also has electronic impulses which are controlled by (or as) thought, does this mean that thought is also manufacturing? We have designations between different natural sciences (biology, chemistry, physics) and social sciences (sociology, psychology, anthropology, economics, political science, and history), but this way of thinking taken to its extreme suggests that all science is a branch of physics.

Links and ideas

- Letter to the Editor [Re: Ballmer chuckles over Linux woes](#)
- [TOOL: The Open Opinion Layer](#) by Hassan Masum

A suggested focus

I sent out a press release³⁰ on March 1, 2004 announcing a ‘Make it legal: don't litigate, use creative licensing’ campaign to encourage software authors, musicians, and other creators of works of the mind to use Free/Libre and Open Source Software (FLOSS) and Creative Commons licensing. This is an extension of the "Stay Legal – Use Free Software" campaign launched two years earlier. Since this press release I received email suggesting that I try to explain this campaign as part of this presentation/workshop.

³⁰ “Make it legal: don' litigate, use creative licensing” campaign <http://www.flora.ca/makelegal200403.shtml> (Accessed March 3, 2004)