Digital Business Ecosystem

Market Watch

Emerging Standards and Practices

Focus on Open Source Software
This edition of Market Watch is aimed at a general understanding of the perspectives of the OSS communities

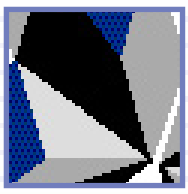
What is OSS
OSS does not have to be free and freely available on the web, but the qualification is made that most OSS is freely available

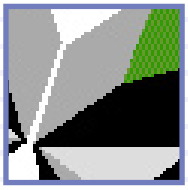
Nature of OSS - how does it work
This section briefly describes aspects of how the OSS community works. It is set within the background of OSS developments since 1990, particularly the development of Linux

November 2004

c o n t e n t s

Market Watch - November 2004



Market Watch

Lead author Peter Stanbridge

Focus on Open Source Software

The second volume of the Market Watch (MW) is dedicated to Open Source software (OSS), a radical movement in software development and distribution that will impact our life far beyond anecdotal criticism of the movement as being a hobby horse of a minority of technical geeks[1].

1. Focus on OSS

The OSS phenomenon is complex. It is difficult to define, partly because its emergence has not been designed, creating therefore, a multitude of communities and practices that cannot be subsumed under a single description.

When discussing OSS, the challenges and implications of legal and licensing far outweigh issues of technology. It is the movement as a social, political, economic and cultural phenomenon that make it difficult for many to comprehend. This edition of MW is aimed at general understanding of these perspectives of the OSS communities, but it is not a technical treatise in the areas of technology, law, licensing, project management and coding and distribution. These are vast topics subject to extensive literature and expert communities. Our aim is, therefore, to provide you with an annotation of literature and other sources on this subject, i.e. to give you an overview of what people are saying about it.

OSS and related subjects such as Free Software can inflame emotions amongst both its champions and antagonists. This study will avoid taking sides with these issues of the protagonists and the resulting political fixations that underpin them.[2] It is of note, but not covered in detail throughout the paper, that Stallman in his address "Free Software: Freedom and Cooperation" makes an important distinction between OSS and Free Software, although the GPL licensing terms were created by Stallman, have often been associated with OSS.

Despite the difficulties defining OSS, the **first section** will provide an overview of OSS that will include some definitions and distinctions. **Section two** outlines the history of OSS (see Annex 2 in this version). This is then followed by **section three**, which outlines how OSS works from the development, distribution and usage perspectives. It draws attention to the

[1] While we have not come across a critique as explicit as this, many of the OSS critiques do come close to writing it off as a hobby incapable of generating commercial strength and stable software, and therefore to compete with genuinely commercial software. Some notable software companies are opposed to OS on the grounds that is unproven and risky. Therefore, so it claimed, the TCO (Total Cost of Ownership) may actually be higher than with proprietary software.

[2] Those that are interested may wish to see "Free Software is a Political Action" – and interview with Richard Stallman found at: http://www.heise.de/bin/tp/issue/r4/dl-artikel2.cgi?artikelnr=6469&mode=html&x=11&y=12 and Stallman et al. 2002, especially chapter 20 "Free Software: Freedom and Cooperation", pgs 155-186.




political and economic aspect of OSS. **Section four** highlights the legal situation primarily with respect to licensing. **Section five** describes how organisations can use OSS. Some case studies are presented as well as an interesting publication designed to help SMEs run their businesses entirely with OSS. The study finishes with **section six**, a presentation of views regarding the future of OSS and implications for the DBE project.

2. What is OSS

The Wikipedia provides the following definition of OSS[3]

> "*OSS means any computer software whose source code is either in the public domain or, more commonly, is copyrighted by one or more persons/entities and distributed under an open-source license such as the GNU General Public License (GPL) (This particular license is often referred to as a copyleft). Such a license may require that the source code be distributed along with the software, and that the source code be freely modifiable, with at most minor restrictions, such as a requirement to preserve the authors 'names and copyright statement in the code.*"

It follows with an explanation that OSS does not have to be free and freely available on the web, but they make the qualification that most OSS is freely available. Notice the explicit mention of the availability of different standard open source licenses governed by different licensing bodies. These will be addressed in section 4.

The "What is" site defines open source in the following way[4]:

> "*In general, open source refers to any program whose source code is made available for use or modification as users or other developers see fit. (Historically, the makers of proprietary software have generally not made source code available.) Open source software is usually developed as a public collaboration and made freely available.*"

Webopedia define OSS with the following[5]:

> "*Generically, open source refers to a program in which the source code is available to the general public for use and/or modification from its original design free of charge, i.e., open. Open source code is typically created as a collaborative effort in which programmers improve upon the code and share the changes within the community. Open source sprouted in the technological community as a response to proprietary software owned by corporations.*"

These encyclopaedias provide an additional definition that refers to the Open Source Initiative Definition discussed next.

3 http://en.wikipedia.org/wiki/Open_source

4 http://whatis.techtarget.com/definition/0,289893,sid9_gci212709,00.html

5 http://www.webopedia.com/TERM/o/open_source.html

Another important source for a definition of OSS is the open source definition published by the open source initiative at www.opensource.org.[6]

The definition starts with a negative. "Open source doesn't just mean access to the source code". This is an understandable start. It is possible that for some people, the meaning of OSS is exhausted by the fact that the source code is provided with the software. But another misconception could be that OSS software is limited to those large global development projects where programmers from around the world contribute to the source code. A type of voluntary socialist software development programme restricted to small number of major well-known products such as Open Office and Linux. But this is where OSS becomes more complex. It can be all these things and much more, but the requirement for providing the source code is just a start.

The Open Software Definition contains ten clauses that define OSS (from the opensource.org perspective[7]):

(1) The first part in the definition states that OSS software can be redistributed (for free or for a fee) as part of an aggregate that may contain software from several different sources.

(2) Access must be given to the source code.

(3) License must be given to allow others to modify the source code and distribute the modified software under the same terms as it was acquired.

(4) A license may restrict the distribution of source code modifications only if it allows the distribution of patch files. This clause helps protect the IC of the original program creators and enables purchases to know who was responsible for the code they are using.

(5) No discrimination is allowed between persons and groups.

(6) No discrimination against fields of endeavour. The license cannot restrict the purpose to which the software is to be used.

(7) The rights attached to a program apply to all to whom the program is redistributed without the need for execution of additional licences. This clause aims at stopping software modification and redistribution rights through other means (such as non-disclosure agreements).

(8) The license must not be specific to a product – "If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution."[8]

(9) License must not restrict other software – if the OSS software is being distributed along with other software, the license for the OSS software cannot place any restrictions on the other software being provided as part of the installation.

(10) OSS licences must be technology neutral. That is, the license cannot limit platform or style of interface.

Another interesting commentary on the OS definition, particularly because the source provides a very comprehensible background to the problems of proprietary software an

---

6 See Open Source Definition http://www.opensource.org/docs/definition_plain.php and annotated version at http://www.opensource.org/docs/definition.php

7 It should be noted that other definitions under different licensing arrangements exist – in fact, these differences account for much of the complexity within the OSS arena. Listing the ten clauses within the opensource.org definition is an excellent place to start an understanding of OSS.

8 Quote from clause 8 in the opensource.org OSS definition. http://www.opensource.org/docs/definition.php




overview history of OSS development, is from Bruce Perens, the treasurer of the Open Source Initiative.[9]

The first point of note in Perens page is his comment that the OSS movement only succeeds as it does on a cooperative and voluntary basis "because of the rights that come with open source". These rights are:

• The right to make copies of the program, and distribute those copies.

• The right to have access to the software's source code, a necessary preliminary before you can change it.

• The right to make improvements to the program.

He then continues "*These rights are important to the software contributor because they keep all contributors at the same level relative to each other. Everyone who wants to is allowed to sell an Open Source program, so prices will be low and development to reach new markets will be rapid. Anyone who invests the time to build knowledge in an Open Source program can support it, and this provides users with the option of providing their own support, or the economy of a number of competing support providers. Any programmer can tailor an Open Source program to specific markets in order to reach new customers. People who do these things aren't compelled to pay royalties or license fees*".

Perens then provides an important justification for this strategy, which at first glance is close to a social ideal. The economics of software is completely different to that of traditional products. This difference in types of capital; between intellectual capital and other types of more traditionally recognised capital has been widely documented in the knowledge management and intellectual capital communities. But software is quite different to many types of service products and intellectual capital in that it can be copied and redistributed for virtually free. Of course, this is the economic incentive that many companies use to apply tightly controlled proprietary licence agreements. It can be cash for nothing. But in the OSS community, software developers who contribute to the OSS projects will only do so if it is not possible for others to profit from their work by taking ownership control of it, and to therefore make disproportionate amounts of money.

The implications for DBE are quite clear. If we do not get the licensing mode right, then we run the risk of a commercial player not associated with the project literally appropriating the work developed by the DBE community under their name and thus taking ownership and control of it via exclusive and proprietary licences.[10]

The OSS definition has been highly influential in the OSS movement, but we will see in section 4 on licensing, that a number of ambiguities exist in the definition that has prompted other OSS licensing bodies to frame their own licensing agreements.

[9] http://perens.com/Articles/OSD.html

[10] Licences for OSS is discussed in section 4 below.

3. Nature of OSS – how does it work

3.1. Introduction

This section briefly describes aspects of how the OSS community works. It is set within the background of OSS developments since 1990, particularly the development of Linux. It will be seen that European developers have played, and still play, a major contributing role in the promotion of OSS and its communities. This section contains subsections outlining the participants in OSS, what is done, how developers collaborate and how disagreements are resolved within the OSS community.

In this section, the emphasis is on the process of OSS and not the artefacts produced. This is an important consideration because processes and methodologies are general enough to travel across industries[11] and are therefore more important than in depth reviews of the artefacts themselves[12].

Information pertaining to the history of OSS is widely available on the web and in many recent publications. The open source initiative provides a brief history of OSS on its website[13] giving particular attention to the OSS formation period of 1997 and 1998. For further information on the early history see the second appendix.

[11] Open source may not only refer to software, there are references in the press to " open source" football team and the term "open source" is beginning to be applied to methodologies and management consulting.

[12] Hence this report provides no in depth discussion on any open source product.

[13] http://www.opensource.org/docs/history.php

3.2. Background in Linux[14]

While it would be simplistic and mistaken to place Linux at the centre of OSS, the development of Linux is a particularly good case study for illustrating how OSS works. It was started by a single person; moved into the OSS area; grew particularly fast; and suffered organisational setbacks that may have killed the project or resulted in major code forking.[15]

Linux is now a major technological and marketing phenomenon, currently running on more than a third of the servers that make up the web.[16] It began with a graduate student of the university of Helsinki named Linus Torvalds. After working on a PC version of Unix (Minix) Torvalds started to build his own operatingsystem kernel around Minix. In autumn 1991, Torvalds let go of the Minix scaffold and released the source code for the kernel of his new operating system, which he called Linux, onto an Internet newsgroup, along with a note requesting help in completing the operating system development.

By the end of that year, nearly 100 people worldwide had joined the newsgroup providing fixes and code improvements.

[14] Based on Weber Chapter 3, 4 and 5. The story is actually so complex and intertwined a reading such as Weber is essential for those interested in OSS history and how the community works in developing software.

[15] Forking is a particular danger in OSS, where different sub-communities break into new communities and develop a software product which "forks" away from the main stream. This happened even in a commercial setting with Unix, within which different software companies provided the own versions, often incompatible with competitor versions.

[16] Weber pg 56




Interestingly, Linux has become a major competitor to Microsoft operating systems even though the Unix dominance has come to an end.

The ride to a mature Linux was not smooth however. Conflicts existed over the nature of the Linux Kernel[17] the differences over Minix[18] development methodologies – get it working and then patch, vs. the grand design and release when perfect,[19] the porting of the kernel from Intel chips to other platforms[20] and the overloaded stress suffered by Torvalds in attempting to incorporate changes. This last point nearly destroyed the Linux development because code contributors were beginning to feel that Torvalds was not and the overloaded stress suffered by Torvalds in attempting to incorporate changes. This last point nearly destroyed the Linux development because code contributors were beginning to feel that Torvalds was not responding appropriately to their improvements.[21] The resolution of this rift is a great testimony to how the community can work together, and how "leaders" who have influence, but no formal position can mediate disputes and pull the development effort towards its goals. Another challenge faced by the Linux development was the integration with proprietary components and proprietary development tools.[22] The rift between the free software purists and those with a more pragmatic (but not necessarily correct) bend over proprietary components grew with the commercial interest in Linux.[23]

---

[17] Whether it should be monolithic or not, it is not currently fashionable to build monolithic kernels today, but Torvalds initially selected this option for OSS development.

[18] Relating to an exchange between Torvalds and Tanenbaum over the kernel. Tovalds comment was "Look who makes money of minix and who gives out Linux for free" (quoted from Weber pg 101).

[19] For example, the rift between van kempen as lead developer on network modules and the rest of the community. Torvalds finally sanctioned a parallel development by Alan Cox of the UK.

[20] Initially the DEC Alpha – this enabled major changes to the way the kernel was coded that would enable Linux to port to multiple platforms.

[21] A similar occurrence happened with the NCSA Web Server. Many of the NCSA developers had moved to Netscape to develop the Netscape proprietary web server. The remaining NCSA developers couldn't cope with the fixes and extensions received via the development community. On the negative side this shows a weakness in the OSS community leadership. There is no one to take full ownership and lead the project with the same mandatory authority that exists inside a typical organisation. The result is forking, which can be very damaging to the product. On the positive side, this situation in NCSA created a fork which eventuated in Apache – an OSS web server with the largest user-base in the world.

[22] An example was the use of Motif, a proprietary programming too kit for graphical interfaces.

[23] Other examples were Pacific HiTech (later named TurboLinux) which packaged free software along with proprietary software that could render Japanese fonts for the Japanese market – but the most extreme case was Caldera that deeply imbedded proprietary software within Linux making it difficult to separate (and possibly breaking license agreements in doing so).

3.3. How software is written

Many software projects are very complex and operating systems are particularly so. How is this complexity managed? The traditional approach relies on traditional scientific management to break tasks into components and ensure a hierarchical organisation to organise and control it. A typical manifestation of this approach is sanctioned by Frederick Brooks "Mythical Man Month".[24] To Brooks, software is complex to the core. He calls it, following Aristotle, the essence in contrast to the accidents of software development which comprise the tools, languages, technical environment etc. It is not possible to penetrate the essence (and thereby simplify it), but the accidents can be controlled through a carefully designed hierarchical software project organisation. Architects design the frameworks for an application and coders' code. Analysts analyse user requirements and throughout the whole process careful project management and measurement ensure the best possible implementation environment. But even in this environment, adding resource by n units increases the code generated by at best n but bugs by n squared. It is clear that this type of Taylorism" has worked very well in the software community despite Brooks pessimistic law (always applicable because it is impossible to crack the problem of essential complexity).[25]

Yet the OSS community completely breaks with this traditional approach to organisation. Given that Brooks's law relating to the mythical man month cannot be magically inapplicable to OSS and given that many of the OSS projects such as Linux are highly complex the OSS community must address their solution to the complexity problem in their unique way of organising.

The following points provide some of the basic aspects of OSS organisation:

1. Voluntary participation

2. Voluntary selection – a person is free to work on whatever aspect of the project takes their interest

3. Labour is distributed, but not divided in the industrial sense. There is no consciously organized for enforced decision on work[26]

4. Typically software development centres on a core product base. Since the source code is freely available, anyone can modify it for their own use.

5. How this modification takes place will vary on the licensing body. For example, BSD-style licences are minimally constraining,[27] whereas the GPL licence is more constraining since it ensures that full OSS rights must pass onto others down the software chain, even if the OSS delivered is just a component.

---

[24] Brooks 1995

[25] Arguably such a view would be generally accepted within the business community simply because software has changed the face of business over the last 50 years, and to a large degree, proprietary software works and is a crucial and indispensible ingredient in business today. Open source is linked to the issue of patents and copyright. As demonstrated in a recent conference in Brussels (Intellectual Property Summit 2004) there are different schools of thought around the degree to which software should be patentable. One school argues that innovation, even at SME level, depends on the degree to which computer implemented inventions are patentable. Others argue the opposite, that patenting is against smaller enterprises and innovation. In this edition we want to take a neutral view. The debate is still very much open on this and the European Commission, as the legislator, is playing a key role.

[26] It should be noted that this point relates to the participation in the development process. The task of accepting software changes into the core application can and does involve a decision making process.

[27] Meaning, for example, that someone can take the open source code and include it in a fully proprietary product.




6. An important part of the process is the feeding back of changes into the core product base. Again, the process will depend on the licensing organisation and the organisation of the project itself. Typically development in a BSD-style product is centred on a small team who may or may not organise to accept changes back from the user-base. They may receive bug-reports and suggestions. In the GPL license, projects typically provide a mechanism for the user base to provide real changes and extensions to the core product base, which are then incorporated back into the product.

7. This process of returning extensions, modifications and bug fixes is often known as check-ins. But in practice, a mechanism is provided for testing, "colleague" peer review and an acceptance process. In the case of Linux, the final decision is made by Torvalds.[28]

8. OSS projects do have project leaders who invite contributions in certain areas. Of course, the role of project leaders in an OSS development is considerably different to a conventional software development project. They have to market their requirements to the community and do not have line authority over those who contribute for them.

3.4. The size and distribution of the OSS community

This is a difficult question and the research efforts to provide metrics and a collection approach are still in their infancy.

The Orbiten Free Software Survey undertaken in 1999 – 2000 identified 3149 discrete open source software projects and 12,706 identifiable developers. Many more hundreds of unidentifiable contributors were found.[29]

The sourceforge website provides on their home page statistics about the software development initiatives hosted by them. As of December 1st 2004, 96,111 projects are registered and 664,010 users registered.

A list of participating contributors and their countries to the Linux development was supplied in the release of version 1.0 of Linux and again for version 2.3.51 (March 2000). The figures for the 2000 release are interesting. Thirty one countries are represented and once adjusted to per capita, the USA with the most absolute number of developers is very low on per-capita contributors. Finland and the Netherlands have the highest number of per-capita contributors.[30]

Weber draws a number of other interesting statistics in his book. For example, in all countries represented, the number of contributors' per-capita has increased steadily

---

[28] It was during this process that the project entered difficulties. Torvalds was failing to incorporate agreed changes fast enough for those using it. The example also illustrates that there exists an important degree of organisational control in the community. The tensions in the Linux case illustrate how effective problems are overcome within the projects (see Weber 2004 Chapter 5). The Apache OSS project uses a governing committee to accept changes into the core product. Tensions in OSS development can cause forking – but this is very rare – another indication of the strength of emergent organisational structures in OSS.

[29] Sited in Weber, 2004, pg 66

[30] Weber, 2004, pg 67

since the start of the project. The conclusion he cautiously draws about the Linux community from the statistics he analyses is that a core of several hundreds of central members do most of the work while several thousand are comparatively peripheral who make small changes and suggestions. Most contributors focus on a narrow specialisation in the overall project.

Weber is cautious about these conclusions because the sample data is still narrow (limited to a few projects) and not always scientifically produced. The metrics are difficult to classify as well. For example, a few lines of code may take a considerable amount of effort if it is addressing a very difficult and core issue, whereas a simple bug fix may involve many lines of code written rather quickly.

3.5. What do OSS developers do

OSS development scrambles the conventional picture of industrial economics. In particular, the concept of producers and consumers is blurred as is the concept of property ownership rights. How do OSS software developers do it? The following eight general principles of the emergence of OSS software development is presented.[31]

3.5.1. Motivation of the OSS contributor

Programmers are highly motivated to contribute toward the project, although there will be many different motives involved. It may be the technical challenge, a political one or an entirely practical one (the project is needed). The project leaders themselves are particularly important in creating motivated volunteers to work on coding problems that may not always be too exciting.

Another important area of motivation is recognition from peers. This is done in a number of ways – from the project team recognition to product release credits. There must be sufficient motivation to engage a developer to seek projects in the open source development community[32] in preference to working in isolation on his own coding challenges.

3.5.2. The practical nature of most OSS development

Most software developed in the OSS community is aimed to solve important issues that are at the heart of commercial software development. Most programmers are motivated by the fact that software will be used and will help them to solve their own software development problems.

3.5.3. Freedom to use previously developed code

With an estimated 75% of software being written for in-house use, a common complaint amongst commercial programmers is that code is frequently re-invented. Some argue that there are good commercial reasons not to rely on externally provided proprietary code in a commercial setting.[33] But the open source community does not have any constraints in this way, and so the OSS community philosophy easily fits with the psychological aspiration of programmers not to re-invent code.[34]

3.5.4. A new method to solve problems

Weber quotes Daniel Hillis[35] from a private conversation as saying that "there are only

---

[31] Based on Weber, 2004, pgs 73 – 93. Weber draws on his own observations and interviews, plus unreferenced analytical work of Eric Raymond.

[32] The largest such community is available at http://sourceforge.net/

[33] This is argued by Oliver Williamson in Williamson 1985.

[34] Unless of course a masterful improvement is possible.

[35] A well known physicist, computer scientist and MIT graduate.




two ways we know of to make extremely complicated things. One is by engineering and the other is evolution". The OSS community works with diversity in parallel with large numbers of contributors. While the project leader may set a problem, the solution is tackled by a diverse community simultaneously and globally. The whole process lacks central control in the traditional management sense. It is important not to create a false bifurcation here – it is not that one is right and the other is wrong. Both approaches have their strengths and weaknesses, the biggest weakness in the evolutionary approach is that there is no guaranteed *a priori* optimal end result and it may take a long time coming. The Engineering approach is fine if the engineers always make the right decision but in practice and in complex situations, this is hardly possible. However, the inefficiencies spoken of in the emergence model is not entirely applicable because the complexity involved in the OSS bears the hallmarks of social complexity, not purely scientific complexity, which does not consider the dynamics of free choice, goal directed activity and human intentionality.[36] As a complex social phenomenon, OSS community is well placed to deliver complex yet high quality applications within short periods of. And practice seems to bear this out.

3.5.5. Different testing paradigm

Paul Vixie makes an interesting case for the superiority of the OSS field testing environment.[37] He states that "The essence of field testing is its lack of rigor". The problem with designed testing is that it cannot consider real experience and can nevercover all the paths the code will execute.[38] OSS has the benefit of a large number of testers, many of whom will test while examining the source code. Power testers can correct bugs as they test. With a large pool of testers made up of different interests and skills, software testing will be more thorough and dynamic. Testers tend to be more sympathetic to the software they are testing when they haven't had to pay money to obtain it, generating a more constructive test environment. With so many strangers testing a piece of software, an open source developer is "kept on his or her toes in a way that no manager or mentor ever could."[39]

3.5.6. Improved documentation

OSS developers need to keep good documentation of their code. The voluntary decentralized distribution of labour could not work with out it. In contrast, many commercial organisations do not see good code documentation as a priority as it doesn't add commercial value to the code being sold. However, Weber (please see Early History section in Appendix 2 for more details on Weber's book) reports that this ideal state does not always obtain even in OSS development. Programmers are frequently in similar time pressures to commercial software developers. During the early days of Unix,

---

[36] See Juarrero 1999, Boisot 1995 (and 2000) and Snowden and Stanbridge 2004.

[37] Vixie 1999.

[38] Unless, as Vixie explains, you are comparing software testing with that of NASA.

[39] Vixie 1999, pg 99.

Dennis Ritchie established a set of principles for software documentation that has spread throughout the open source community.[40]

3.5.7. Release early and release often

As part of the evolutionary processes inherent in OSS, a process of releasing frequently and early is a healthy approach to ensure that the software converges quickly to a mature and stable end. Interestingly, there is nothing guaranteed in this process, since frequent change can overwhelm the system and kill it. That this doesn't happen in OSS development is an interesting phenomenon that warrants study.

3.5.8. Keeping the communication channels open

OSS developers talk a lot – creating discussions around specific technical problems, general issues and problem solving. They talk about new ideas, old bugs, new hardware – about almost anything. The community encourages free speech, which can at times get quite heated and opinionated. But free and frequent speech has a vital part to play in the dynamics of OSS development and the fast and free movement of ideas and knowledge.

4. Licensing and legal[41]

4.1. Definitions, licences and property rights

This has developed into the most complex area of the OSS movement. As an opening remark to this section no-one on the DBE Market Watch team is a qualified lawyer and views expressed are those of leading experts in the field. Most of the research publications into OSS licensing (including work on Patents and Trade Marks) are produced by lawyers. Given the nature of OSS, one would be led to think that this shouldn't be so complex. Surely it is the avoidance of intellectual property rights that define the OSS community? But this in fact isn't true. OSS developers vehemently uphold copy right and intellectual property rights – and one of the most important reasons is to ensure that those who pass on OSS do so under the same open source licensing as the software they obtained.

Rosen describes the changes in licensing with the advent of OSS license agreements as a revolution.[42]

The first license to be established in the OSS arena was introduced by Richard Stallman in 1989, with his GNU General Public License (GPL) for the UNIX software that he had developed.[43] This was followed in the same year by Bill Joy with his first release of UNIX software under the University of California's Berkeley Software Distribution (BSD). The number of licences available to the OSS software developer has since then mushroomed, although the majority of licenses are registered under either the BSD or GPL (see the appendix for a list of licenses listed in the Open Source Initiative website).

The OSI has provided the open source definition of ten key points that licenses must adhere to in order to be registered under the Open Source Initiative.[44] This is an important stipulation because most open source software portals, such as SourceForge will only accept an OSS hosting if it is licensed under an OSI approved license agreement.

Open source is build upon a foundation of intellectual property law, particularly copyright law, but also patent and trademark law.

---

40 Weber, 2004, pg 79.

41 References for this section are Rosen 2005, St Laurent 2004 and van Wendel 2003.

42 Rosen 2005.

43 Stallman does not like the association of GPL with OSS – he regards his distinction of Free Software as important and denies that the idea of "freedom" captures his idea of free – see Stallman et al. pg 156.

44 These ten points were presented in section 2 above.




This law states that "OSS is owned by its authors, who license it to the public under generous terms".[45]

OSS licensing can be classified in a number of ways. Rosen provides a useful classification beginning with the academic licenses (BSD, MIT, Apache, Artistic and Apache V2.0) then following with the GPL, MPL, CPL and OSL licenses in turn.

St. Laurent also covers a number of proprietary and semi-proprietary licenses such as the Sun Community Source License, the Commercial Use Supplement and the Microsoft Shared Source Initiative.[46]

Van Wendel et al.[47] provide a section outlining the role of Governments in balancing the software industry between proprietary and the commons.[48] An assumption made quite explicitly by the authors in this book is that both communities have to work in balance within the software development environment. The strengths and weaknesses of each balance out if the software development community as a whole is in some form of equilibrium between the two – and it is the role of governments to ensure that this balance is kept (and the book explains how governments can do this through the administration of licensing laws). They provide three scenarios – in balance, business dominates and commons dominates. In only the first scenario the authors see a healthy software production environment. They argue that only in such a balance "encourages both uncertainties for innovation and the effects of open communities for speed and quality". The assumption made here, is that only proprietary licences will generate sufficient capital across the whole industry to finance innovation.

They provide the following table to indicate actions available to governments to ensure that this balance is kept.

	Balance	Business Dominates	Commons Dominates
Position of commercial IP	Coexistence	Businesses colonize IP in the commons	Viral licences undermine proprietary IP
Concern of Governments	Checks and balances	Future of the commons	The corporate model and the future of R&D
Role of Governments	Monitor stability	Limit scope and strength of patents and copyrights	Extend and enforce software patents and copyright
Concern of open source licences	Passive	Codify licences in law	Provide the means to enable the commons to be colonized

The book was based on public funded research in the Netherlands and could be quite influential. The case for supporting a balance between proprietary and commons software has been made and it is the opinion of this author that the language presented in the table is quite dangerous.

The most severe situation that can occur within the OSS community is a serious

[45] Rosen 2005, pg xx.

[46] Although these licenses should not distract attention away from the vast differences these licences have over the truly OSS licenses. But they do form part of the economy of software development outside the full constraints of classical proprietary licensing.

[47] Van Wendel et al. 2003, pg 113 - 116.

[48] Another name for the OSS and free software community.

licensing litigation case. The most serious to be heard recently is the SCO case. In this case, the company that produced SCO Unix made claims of copyright over sections of code in the Linux operating system. This case has generated an absolutely huge amount of comment and press space, and Microsoft, who has shown a strong interest in precedence set by the SCO case, has taken a hard line on the issue of patent infringement. For example, Microsoft CEO Steve Ballmer told Microsoft's Asian Government Leaders Forum that "Linux violates more than 228 patents and that Asian countries entering the WTO will be sued for the copyright infringements in Linux".[49] It should be noted that the only case to proceed to court – the Daimler Chrysler case, had the SCO claims rejected.

Why has the OSS community created such a proliferation of licenses? To understand this, the legal status of the open source definition is discussed followed by the major motivations of the most popular licenses.

Rosen[50] highlights some difficulties with the ten criteria provided by the OSI. He claims that the open source definition has created some confusion, particularly in the vagueness and inconsistency in the usage of expressions. For example, Rosen mentions the phrase must allow as it occurs in the third criteria.

> *"The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software".*

Rosen draws our attention to the different uses of this phrase in the passage quoted. The first usage is a *must allow* while the second is a *may require*.

The *shall not require a royalty* in the first definition has also created confusion. OSS is not necessarily free software, but you never have to pay a royalty or license fee for the right to make copies. The use of the word *discrimination* has many unintended meanings that may read against the OSS case. For example, that it discriminates against proprietary software rights. Rosen draws out several other examples throughout his discussion of the various license agreements.

4.2. The Academic licences

One of the first licenses for OSS was the BSD (Berkeley Software Distribution). This license is very short, simple and liberal, using the idea of the freedom associated with academic learning, where teachers are encouraged to publish rather than hide their results and students are encouraged to apply and improve on their learning in their area of work.

The BSD license had a number of protection objectives for the university, but basically, it enabled the freedom to use and modify the software in anyway the user felt fit. The consequence has been a huge amount of software developed under this license in response to the initial software provision (people putting back more for what they have taken). There is no obligation in the license to pass on software developed under this license using the same conditions in which it was obtained – in other words, it is possible to wrap BSD software into a completely proprietary package.

The MIT created their own version of the BSD license and made their version tighter and simpler to read. But the MIT version still does not restrict downstream licences.

The Apache license is also an academic license. "This means that Apache software

---

[49] Quoted from http://www.theregister.co.uk at http://www.theregister.co.uk/2004/11/18/ballmer_linux_lawsuits/

[50] Rosen, 2005, pg 6 and 7




may be used by anyone, anywhere, for any purpose, including for inclusion in proprietary derivative works".[51]

4.3. The GPL

The GPL is an important license because it has been highly influential in creating a large commons of software that is freely available to everyone world wide. Importantly, the GPL license agreement has stopped much software from being captured by proprietary software interests and converted into restricted private property.

The key to the GPL is reciprocity. Rosen puts it: "You may have this free software on condition that any derivative works that you create from it and distribute must be licensed to all under the same license".[52] It is the clause in the GPL expressing this idea that has created the most passion in its adherents and its detractors. The important detractors are those from the academic licensing world, who can place their products into a GPL licensed product but not visa versa. The implication being that the GPL has created an Island. Rosen's explanation of these different communities is that the academic licenses are for generous donors of software and the GPL for generous sharers. The effect is that we have two public commons of free software and not one.

An important term often associated with the GPL is "copyleft". It is a play on words from copyright and is meant to highlight the differences between the two. In particular, the GPL license is designed to grow the public commons of software rather than to have each owners copyright to pull from it. But copyleft is an important concept because it relates to reciprocity, a central concern for the OSS community. The following points illustrate the key aspects of the GPL licence.[53]

Preamble: The authors of the GPL, Richard Stallman and Eben Moglen, make their important claim as to the principle of OSS – "The GNU General Public License is intended to guarantee your freedom to share and change free software – to make sure the software is free for all users". They then continue with their definition of free software. "When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free soft-ware (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things".

The idea of the GPL is then to ensure that know one can ever deny the recipient of any of these rights.

Reciprocity obligations: Licensees must use the GPL as their license if they distribute modified versions of the software. Any resulting derivative works will also be free software. This obligation is in place to ensure that middlemen cannot effectively

[51] Rosen 2005, pg 91.

[52] Rosen 2005, pg 103

[53] From Rosen, 2005, Ch. 6. It should be noted that Rosen comments on some of the vague and imprecise language of the GPL.

hijack the OSS software and re-license in their own terms.

This obligation also provides safeguards against the type of software incompatibilities experienced in the Unix community, where different vendors licensed their own versions that were incompatible with other versions.

GPL as a template: The GLP can be applied without the software author having to create a specific software license containing the name of the software. This is achieved by stating a notice in his source code that the software may be distributed under the terms of this General Public License.

It is possible, however, to make an explicit reference to a particular version of the GPL. Without this reference, the licensing conditions will change as the GPL changes.

Problems on linked, collective and derivative works: The GPL is not clear on many areas relating to derived works, collective works and linking. The use of "derivative" within the GPL is also ambiguous between collective and derivative. But in the normal copyright sense of the word, these are important distinctions. A derivative is based on a previously existing work (via modifications, for example) and a collective being the combination of components, which may have different licensing arrangements. Not withstanding the GPL license itself (which appears to include collective works and derivative works within its license), Rosen expects that in a court of law, only derivative works will be treated as coming under the terms of the license.[54] The implications for DBE are extremely important, because the run-time environment will contain large collections of collective works that are linked and imbedded. Careful legal advice is advised in such situations, particularly if the project needs to be precise about license implications to software providers to the DBE and their users. The *Lesser General Public License* LGPL addresses the question of linking software. In fact, the GPL recommends the use of LGPL in those situations where an OSS program is a component that the author may wish users to link from proprietary sources.

Grant of License: The GPL enables people to copy and distribute verbatim copies of the program's source doe as you receive it, in any medium provided (GPL section 1). It then grants the user to modify the Program or any portion of it, thus forming a work based on the Program. The person is then free to distribute the program or a modified version of it under these same terms.

Access to Source Code: The GPL allows licensees to copy and distribute the source code. The license is worded to ensure that usable source code is available for licensed software. (and it is normal practice to provide source for an entire work being distributed, not just the sections being written by the distributor).

At no Charge: With regard to distribution of derivative works, the GPL reads: "You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this license". This is to insist that licensee's derived works are distributed at zero cost as the licensor had done – and that the derived work is to be treated as a whole.[55]

The GPL and Patents: The GPL is unclear on patents. A provision in section 7 states that if a Program licensed under GPL falls under a patent judgement and can no longer be

[54] See Rosen, 2005, pages 113 – 120.

[55] There is a provision in GPL section 1 for the recovery of distribution costs (although again, this section is ambiguous as to what costs can be allocated).




freely distributed under the GPL license. The key point in the provision is that the mere threat of patent infringement is not sufficient to bring about a change to the license. A case has to be brought and settled prior to any changes taking effect.

4.4. The Mozilla Public License (MPL)

This license was created by Netscape when it realised its Internet browser to the open source community. A key requirement for Netscape was to ensure that modifications and improvements to the software were returned to the community. They also had some misgivings about the GPL license. Firstly, they had obligations to other software vendors software used in the original browser. They were also unsure on the implications for other Netscape software that wasn't being released to OSS. The result was the MPL written by Mitchell Baker, an attorney at Netscape. This license has formed the model for subsequent commercial open source licences that followed. Rosen describes this license as "a high-quality, professional legal accomplishment in a commercial setting".[56]

4.5. The Common Public License (CPL)

IBM created the CPL as a license template for other companies to use; companies who were distributing software that may be useful to IBM and others to use or to sell with no ambiguous license provisions hanging over them.[57] The CPL is another very tight and professional contract that has its key termsdefined and unambiguous. It is also fully compatible with the OSD although Rosen notes some complexity and uncertainty surrounding any patent rights within software code.

The CPL, like the other licensees, has reciprocity obligations, but the CPL has an interesting exception – that is where a piece of software is in a separate module and it is not a derivative work of the program. This type of exception provides a high degree of flexibility when OSS components are used within a larger piece of software that may be registered through a different licensing scheme.

4.6. OSL and AFL licenses

This section will close with a very brief statement of these two licenses, which were written by Lawrence Rosen[58] in response to the vagueness in the academic and GPL licenses. The Open Software License (OSL) is a reciprocal license (like the GPL) and the Academic Free License (AFL) is an academic license. The two licenses have been developed side by side, so they share the same terminology and clauses in their common elements. The differences related to the reciprocal clauses in the OSL.

Apart from the careful and common wording, Rosen claims the contracts are also simple and short.

5. Use

This section is a shorter section designed to provide some case study evidence for the strength of OSS as well as an examination of the use of OSS in SMEs.

---

[56] Rosen 2005, pg 142.

[57] Rosen 2005 pg 161.

[58] Rosen 2005, pg 181.

5.1. OSS for SMEs

One argument raised against OSS is that it is too complex to use for many SMEs who may not have the ICT skills. As a firm-wide installation will require a large number of separate software installations, management and configuration of these components will be beyond many SMEs.[59]

This is a compelling argument that could imply that for SMEs, the cost of ownership of OSS may be higher than a proprietary environment such as a Microsoft suite of products.

One problem with this argument is that it assumes that a proprietary offering from a company like Microsoft is itself easy to install, configure and manage. While this may be true for a desktop Windows XP and Office, it is not true of the more advanced software features such as networking, web servers and Internet software. In addition, many packages will need to be purchased and installed to cover the full range of business functions required by the upper half of the SME market.

Either way, some form of professional ICT support or help may be required. For those with a degree of ICT skills, help is available such as John Locke's OSS book[60] "Open Source Solutions for Small Business Problems". This book provides a remarkable coverage of various OSS options for a small business and provides sufficient help to install, configure and manage them. To gain an insight into the broad range of OSS software available to small firms, the book provides detailed instructions of OSS to support:

1. office network
2. desktop
3. e-mail server
4. web server
5. CRM
6. calendar and schedule management
7. document management
8. financial management
9. resource and project management
10. e-business
11. marketing
12. remote connection
13. privacy
14. security
15. wireless connection
16. disaster recovery
17. privacy and protection

This is a significant list of business functions supported by OSS, and with the use of Linux, the comparison of cost of ownership must be considerably less than with proprietary software (although no actual financial comparisons have been sited).

5.2. Open software trials in Governments

The Office of Government Commerce (OGC) announced in September 2003 that it would coordinate a trial of OSS products within a range of public bodies. This trial was undertaken in conjunction with IBM and Sun Microsystems.

The project trials obtained customer experience on the following key issues:

- Is OSS a viable alternative to proprietary software
- What factors inhibit the adoption of OSS
- What evidence exists for the financial benefits of OSS
- Lessons learned in planning, implementation and operation of OSS

---

[59] See, for example, "Case against open source" at http://uk.builder.com/programming/windows/0,39026618,20265682,00.htm

[60] Locke, 2004.




The project reported the following summary key conclusion on these points

- OSS is viable and credible
- Desktop applications were the main inhibitors for OSS adoption. File conversion as well as finding suitably sophisticated applications were the two main problematic areas. No inhibitors were found for infrastructure software
- Evidence was found for significant savings in hardware, software and infrastructure costs.
- Desktop conversion requires meticulous planning and training as well as detailed migration planning.

The report contains detailed conclusions that cover many of the standard benefits (such as lack of lock in, quick turnaround of bug fixes and software updates) as well as a catalogue of perceived weaknesses. A summary of the main perceived weakness presented are:

- Uncertainty as to exactly what OSS is
- Misunderstandings of the licensing and IPR implications of using and purchasing OSS
- Difficulty in identifying appropriate OSS applications – particularly for business applications
- Lack of quality end user documentation
- Lack of experience in converting to OSS environment

Finally, the report provided some interesting case studies. The scope of savings can be indicated by one local body that were able to reduce the number of servers from 60 to 10.[61]

[61] Moving from Windows NT based OS to Linux and apache (f or their web servers).

The MOD claimed that their OSS environment could be made more secure than their proprietary one.

6. Future and implications

Steven Weber is a political scientist, not a computer scientist. His book "The Success of Open Source" is an important publication because while it covers many of the standard features of OSS (such as licenses, definitions and history) its bulk is dedicated to examining the OSS phenomenon as a political, cultural and economic phenomenon. It is clear that from a political scientist perspective, Weber considers that the OSS movement has created a new economic force that will change the balance of power traditionally associated with software copyright ownership for ever.

Weber's significant terms are more embracing than what would typify a book covering a mere technology innovation. For example,[62] he states that "open source is a way or organizing production, of making things jointly". He then indicates that it isn't easy for human beings to produce complex integrated systems. The traditional solution is property ownership, hierarchy, divisions of labour, reduction of transaction costs and so on. But "the success of open source demonstrates the importance of a fundamentally different solution, built on top of an unconventional understanding of property rights configured around distribution". But this alone would not be sufficient. Weber claims that "open sources uses that concept (the new understanding of property rights) to tap into a broad range of human motivations and emotions, beyond the straightforward calculations of salary for labour". And while

[62] All quotes in this paragraph are from Weber 2004, pgs 224 - 225

many have charged the community as chaotic, he continues "…and it relies on a set of organisational structures to coordinate behaviour around the problem of managing distributed innovation, which is different from division of labour." Weber is sure that these concepts have broad consequences for economic and politics.

OSS is making us think differently about property. Given the IPR concerns of the OSS community, it is quite shocking for traditional thinking to see these rights expressed in terms of freedom to distribute and contribute instead of the traditional concept of property as a right to exclude. When one considers that the modern concept of property and property rights has been partly responsible for the growth of wealth in those countries that implement the traditional view, a challenge in any significant economic sector to this traditional view could have extreme consequences to wealth distribution. Weber even compares the phenomenon of the OSS community to terrorist cell networks. While the USA (and other countries) struggle to determine an effective approach against terror networks (because they are so different to traditional enemies and they behave differently) traditional software companies will struggle to compete against OSS (which may indicate why some of the major ICT companies, such as IBM, Sun and HP have embraced or invested in OSS).

But an important consideration in the debate is that there is no true or natural state in the manner in which we understand property. Property is a constructed concept and can be construed in different ways.[63] In this sense, to change the construction of property means to change the way in which the network of human relationships will work – and the OSS view is highly networked in which case the results will be more emergent than designed.

63 Weber 2004, pg 228

Weber then provides characteristics shared between the Internet and the OSS community. These include the networked effect discussed above. In addition, low cost of entry, low transaction costs, removal of central control.[64] The removal of central control could turn out to be the OSS community's greatest power. In particular, the OSS approach may provide the needed acceleration in software development that is required to match the pace in hardware development, which has traditionally well out-paced software development.

But there are some inherent weaknesses in this model for the future too. One is that the community could get locked into debugging and developing a single platform whereas it should be investing in a completely new paradigm. A second potential problem is that interest could wane once the major OSS products become more stable and main-stream. While this may provide creative impetus for new products, commercial users need to have continued product support over reasonably lengthy periods of time.

This section will close with a quote from the Openz website. After explaining that the OSS movement had made the Internet possible – that is, without OSS, the Internet would not have been possible, they continue:

> "*But the Internet does exist, and it has proven its robustness. Despite all efforts of the world's largest companies to stifle its openness, the communities surrounding free software have not only survived, they have flourished. Although some of the high profile companies who have*

64 Here a comparison between the Internet and a telephone network is interesting. The telephone network can also create new worlds as it can generate new networks of communicating people, but there are limitations because the network is centrally controlled. OSS is more like the Internet which is controlled at the periphery. This creates a much more fluid, dynamic and faster changing environment not possible in a centrally controlled network.




tried to build businesses around open source software in an attempt to vy with the big closed source corporates have failed spectacularly, many of the smaller, faster, more agile ones have thrived quietly, as mammals did among the dinosaurs just before their extinction... For evidence of this, one need only survey the activity on one of the open source world's greatest assets, Source Forge created by VA Linux Systems before they, too, succumbed to the blight of the open-source-company-emulating-a-corporate, rebranding as the more ideologically ambiguous VA Software. Luckily, SourceForge has survived the throes of business. The sheer number of projects, managed voluntarily by software developers at non-software focused companies and institutions, or in the smaller more agile open source focused companies mentioned previously, and the rate at which that number is growing, is a testament to the turning tide and the growing momentum behind the global open source movement.".

7. References

7.1. Web References

7.1.1. Open Source Initiative

Main website: http://www.opensource.org

OSS Definition with annotation: http://www.opensource.org/docs/definition_plain.php

OSS Definition without annotation: http://www.opensource.org/docs/definition.php

OSS History: http://www.opensource.org/docs/history.php

7.1.2. Other references

Webopedia Definition: http://www.webopedia.com/TERM/o/open_source.html

What is definition: http://whatis.techtarget.com/definition/0,289893,sid9_gci212709,00.html

Wikipedia definition: http://en.wikipedia.org/wiki/Open_source

Peren's definition: http://perens.com/Articles/OSD.html

Debian Free Software Guidelines: http://www.debian.org/social_contract.html#guidelines

Open knowledge history: http://www.open-knowledge.org/writing/open-source/scb/brief-open-source-history.html

Openz history: http://www.openz.org/oshistory.php?umsSession=af1299c4f33063cc8f53b90a094005aa

Netaction history: http://www.netaction.org/opensrc/future/

An OSS development website http://sourceforge.net/

A case against open source http://uk.builder.com/programming/windows/0,39026618,20265682,00.htm

7.2. Books and Published Papers

Boisot Max, 1995, *Information Space: A Framework for Learning in Organizations, Institutions and Culture*, Routledge

Boisot Max, 2000, *Knowledge Assets*, Oxford University Press

Brooks Frederick P. 1995, *The Mythical Man Month: Essays in Software Engineering*, Addison Wesley, 2nd Edition.

DiBona Chris, Ockman Sam, Stone Mark, 1999, *Open Sources: Voices from the Open Source Revolution*, O'Reilly.

Juarrero Alicia, 1999, *Dynamics in Action: Intentional Behavior as a Complex System*, MIT

Locke John, 2004, *Open Source Solutions for Small Business Problems*, Charles River Media

Rosen Lawrence, 2005, *Open Source Licensing: Software Freedom and Intellectual Property Law*, Prentice Hall

Snowden Dave, Stanbridge Peter, 2004, "The Landscape of Management: Creating the Context for Understanding Social Complexity" in Emergence Vol 6 numbers 1 & 2. (currently accessible at http://emergence.org/ECO_site/web-content/ECO_6_1-2.html)

St. Laurent Andrew M, 2004, *Understanding Open Source & Free Software Licensing*, O'Reilly UK.

Stallman Richard M, Gay Joshua, Lessig Lawrence, 2002, *Free Software, Free Society: Selected Essays of Richard M. Stallman*, GNU Press

Vixie Paul, 1999, "Software Engineering" in DiBona et al. 1999, pgs 91 – 100.

Weber Steven, 2004, *The Success of Open Source*, Harvard.

Van Wendel De Joode Ruben, De Brujin J. A., vab Eeten Michel J.G., 2003, *Protecting the Virtual Commons: Self-organising Open Source and Free Software Communities*, Cambridge University Press

Williamson Oliver, 1985, *The Economic Institutions of Capitalism: Firms, Markets, Relational Contracting*, Free Press

8 Appendix 1 – Licence

The following list contains the licences available for OSS – this list has been taken from the Open Source Initiative site – http://www.opensource.org/site_index.php.

The list is included here to provide an indication of the complexity of dealing with license issues.

Licenses:

- The GNU General Public License (GPL)
- The GNU Library or "Lesser" GPL (LGPL)
- The BSD license
- The MIT license
- The Artistic license
- The Mozilla Public License v. 1.0 (MPL)
- The Qt Public License (QPL)
- The IBM Public License
- The MITRE Collaborative Virtual Workspace License (CVW License)
- The Ricoh Source Code Public License
- The Python license (CNRI Python License)
- The Python Software Foundation License
- The Apache Software License
- The Vovida Software License v. 1.0
- The Sun Industry Standards Source License (SISSL)
- The Intel Open Source License
- The Mozilla Public License 1.1 (MPL 1.1)
- The Jabber Open Source License
- The Nokia Open Source License (NOKOS License) Version 1.0a
- The Sleepycat License
- The Nethack License
- The Common Public License
- The Apple Public Source License
- The X.Net License
- The Sun Public License
- The Eiffel Forums License V.1
- The W3C License
- The Motosoto License
- The Open Group Test Suite License
- The Zope Public License
- The zlib/libpng License
- The Academic Free License
- The Attribution Assurance License
- The Open Software License
- The OCLC Research Public License 2.0
- The wxWindows Library License




• The Sybase Open Watcom Public License 1.0
• The Eiffel Forum License V2.0
• The Naumen Public License
• The Historical Permission Notice and Disclaimer
• The RealNetworks Public Source License V1.0
• The Reciprocal Public License
• The University of Illinois/NCSA Open Source License
• The Entessa Public License
• The Plan 9 Open Source License
• The PHP License (v3.0)
• The Frameworx License
• The Apache License, Version 2.0
• The CUA Office Public License Version 1.0
• The EU DataGrid Software License
• The Fair License
• The Lucent Public License Version 1.02
• The Eclipse Public License
• The Nasa Open Source Agreement

9. Appendix 2 - Early History of OSS

The history of OSS is very interesting, which is discussed in this appendex.The following points are of interest during that period:

• Its history lies in Unix and free software and the "hacker culture".[65]

[65] It is interesting to note that the open source initiative keep a pure concept of hacker – a hacker is a shared culture community or expert programmers that have built Unix, the Internet and run Usenet and make the WWW work – nothing to do with smashing secure computer systems. See http://www.catb.org/~esr/faqs/hacker-howto.html which is linked from the open source initiative history page.

• The term "open source" came out of a strategy meeting held at Palo Alto in 1998[66] based on the decision by Netscape to release the source of its browser.

• The members of the meeting were attempting to remove the confrontational aspect of the term "free software" and were keen to make a commercial case for OSS. It is during this meeting that the term OSS was coined.

• At an earlier meeting in 1997, the OSD was derived from the Debian Free Software Guidelines[67] initially drafted by Bruce Perins and modified over a number of weeks.[68] [69]

• The Open Source Initiative is now a California public benefit (not-for-profit) corporation.

• Netscape announced its release of the browser source on 22 Jan 1998.

• Debates raged during February 1998 within the hacker community on "open source" vs. "free software". The debates indicate the wider issues of the communities relationship to the business world.

• On March 31st, the source for Netscape Navigator is released. Within hours fixes and enhancements were received off the Net.

• By April, the open vs. free debate is winding down, and the open source community is becoming more broadly known and consolidated.

[66] The people present are evidently significant. They were Todd Anderson, Chris Peterson, John "maddog" Hall, Larry Augustin, Sam Ockman and Eric Raymond.

[67] See http://www.debian.org/social_contract.html#guidelines

[68] The 10th point is added sometime later

[69] For reasons described in section 5 below, the Netscape license could not use the OSS Definition.

• Over this year, a number of large software companies begin to support OSS. Corel, with Word Perfect, IBM using Apache in WebSphere, Sun Microsystems joins Unix – and a number of proprietary software vendors port their products for Linux (such as IBM, HP, Apple, SGI and SAP). Commercial use of Linux is growing.

• In November of 1998, Microsoft's Halloween documents are published, outlining their dirty tricks response to OSS (Linux in particular) which ignites a week-long furore in the national media.

But open source certainly had earlier origins (in the free source movement alluded to above). Many websites provide detail of this history.[70] The stories do vary somewhat. A very detailed and interesting description of the early OSS is to be found at the Openknowledge website, which provides a detailed description of the ups and downs of the early free software movement. It recognises firstly, that it began in the hacker culture of the US Computer Science Laboratories in the 1960s and 1970s. Most of the early community was focussed on building tools for the DEC PDP-10, but the community broke down in the early 80s once the machine had been discontinued (at MIT) and DEC subsequently become heavily proprietary.[71] Other problems for the community emerged as many of the hackers formed private companies to build and market proprietary software or were employed by such companies. One of the key figures in this movement at this time was Richard Stallman who "started the GNU project, to protect and foster the development of free software. A stated goal of the project was to develop an entire operating system and complete sets of utilities under a free and open license so that no one would ever have to pay for software again".[72] Stallman decided to devote himself to creating free software, which meant that every user has the right to:

1. Run the program for any purpose

2. Modify the program to meet their own needs and to redistribute copies

3. Distribute modified versions of the program.

In 1985, Stallman created the Free Software Foundation, a tax exempt charity, to support his work and that of his collaborators. Stallman personally created an enormous body of software: the GNU operating system (based on Unix) GCC (C compiler), GDB (debugger), Emacs (text editor), and a number of other tools.[73]

Openknowledge also acknowledge other open movements of the same era. These include the X windowing system and Perl.

The story then takes up where the Open Source Initiative story starts.

Netaction provides a slightly different perspective on the history, focussing on the technologies used at MIT in the early days of the hacker movement (such as ARPANET) and its progress to the Internet.

Openz have a very broad perspective of the history of open source – especially as it previously occurred outside the OSS domain. It is also very witty in its style. For example, they claim that the scientific community has always been open source.[74] Prior to the scientists there were the

---

[70] The following links being discussed here http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html, http://www.openz.org/oshistory.php?umsSession=af1299c4f33063cc8f53b90a094005aa and http://www.netaction.org/open-src/future/ - a search on Google will find hundreds of others.

[71] And the software built by the hacker teams couldn't be ported.

[72] See the "Contributors " section on the O'Reilly Safari Bookshelf found at http://safari.oreilly.com/?XmlId=1-56592-582-3 for the book DiBona et al. (eds) 1999

[73] See http://www.heise.de/tp/r4/artikel/6/6469/1.html for an interesting interview with Stallman.

[74] Although with industry investing heavily in university research today, it would be interesting to see to what extent this ideal is being compromised.

artisans, "where enthusiastic artisans have always passed on their lore to new generations in an effort to create a positive legacy for themselves". With regard to the early history of OSS, they write:

> "*This idea of building on the accepted foundations of knowledge, the basis of scientific culture, naturally passed into the ethos of the computing world with the advent of the first computer networks in the 60's and 70's where the first young software "hackers" - a self-deprecating reference to "hack writers" designed to downplay the cleverness of their early software endeavours - reached out of their lonely, usually secluded and windowless labs at universities and government laboratories sprinkled around the world to fashion the first virtual communities.*"

There is explicit reference to the continuity between the early work of the hacker community and the development of the Internet. They describe how this community, to the disbelief of the "darlings of wall street",[75] started to build software to be distributed freely. These were mainly people who were developing software in institutions that were not specifically software companies. The interchange of ideas and source code modifications through the news groups and bulletin boards provided high performance software at much cheaper prices than that provided in the commercial software market. Many of the products developed in this way have found their way into the backbone of the Internet. To quote "If it wasn't for the steadfast refusal of Berners-Lee and other champions of open standards to cave in to commercial domination of the standards process - creating closed standards controlled by only one company - the Internet would never have been achievable."

Steve Weber's book "The Success of Open Source"[76] is an excellent resource for a sociological and political perspective on OSS. His chapter on history is very comprehensive and readable. His history begins with PACT (the Project for the Advancement of Coding Techniques) which was a consortium of companies (mainly software engineers from Lockheed, Douglas, RAND and other defence contractors) to build compilers and tools for IBM computers) and the cultural and legal interpretations of a consent decree given by the Eisenhower administration in 1956 restricting telecommunication companies from extending their business outside the telephone and telegraph areas. The interpretation of this decree was to have a huge impact on the OSS movement, particularly as Unix was developed within the Bell Labs. The next phase of development was the introduction of low priced computers from DEC. His history then moves to the MIT work on multi-tasking operating systems, the development and spread of Unix, early predecessors of the Internet and the down period in the 80s. This includes (in addition to the problems described by Openknowledge above) the attempt by AT&T having recognised the value of Unix, to restrict its licences (contrary to their previous interpretation of the earlier consent decree). The story is interesting as proprietary AT&T (and at that time SUN) battled it with Unix based around BSD code (from MIT) supported by the Open Source Foundation. At this point, Unix was in a mess, despite it surpassingly being a multi-platform system. The recession of the early 1990s (where a number of major computing companies either nearly failed or actually failed) could have put and end to this movement, but the development of the Internet created a new environment for OSS to flourish.

---

[75] The very successful proprietary software companies.

[76] Weber, 2004

An interesting episode in this history occurred in the mid 70s when hobby-computers were a rage. The Altair computer was very popular amongst the home hackers, but the computer required Bill Gates and Paul Allen's BASIC computer language. But the hackers were distributing copies amongst each other for free. Gates wrote an open letter to these hobbyists which stated "As the majority of hobbyists must be aware, most of you steal your software". He states that this practice of "thieving" software, in his view, would stifle innovation. "Who can afford to do professional work for nothing? What hobbyist can put three man-years into programming, finding all the bugs, documenting his product, and distributing for free?"[77] Obviously, the hobbyists had ideas of their own, for which our modern computing environment is a reflection of the two worlds, each claiming to be self-evident, but neither in fact being of blind historical necessity.

This takes the history of OSS to about 1990 and the development of the Internet, Linux and other important OSS developments. The development through the 1990s will be highlighted throughout the next section, where a description is given on how the OSS community works.

---

[77] Quoted in Weber, 2004, page 36, 37.