

ITLED4240 Compendium Spring 2012:

Open Source, Open Collaboration and Innovation



Note no
Editors

Date

DART/01/2012
Wolfgang Leister
Nils Christophersen
March 1, 2012

The authors

Wolfgang Leister, assisting research director at Norsk Regnesentral, received the Dipl.-Inform. degree in 1986, and the Dr. rer.nat. degree in 1991, both from the Universität Karlsruhe, Germany. His research interests cover multimedia, computer graphics, computer and sensor networks, mobile systems, health care applications, and free software.

Nils Christophersen is professor at the Department of Informatics at the University of Oslo. He received his MS in cybernetics in 1972 and his PhD in informatics in 1983, both from the University of Oslo. His current interests cover computational life science and bioinformatics, free software and innovation, and Commons-Based Peer Production.

Arne-Kristian Groven, senior research scientist at Norsk Regnesentral. He received his MS in computer science from the University of Oslo in 1991. He has since been working at the University of Oslo and the Halden Reactor Project before joining Norsk Regnesentral in 1996. His research interest includes semantic technologies, security and trust, quality assessments, long term preservation of digital material, Voice over IP, and free and open source software.

Frontpage Image

“Skiltskog på vei til Vestmarkseter”; acrylic painting by Wolfgang Leister; © 1995, image licensed CC BY-NC-ND.

Norwegian Computing Center

Norsk Regnesentral (Norwegian Computing Center, NR) is a private, independent, non-profit foundation established in 1952. NR carries out contract research and development projects in the areas of information and communication technology and applied statistical modeling. The clients are a broad range of industrial, commercial and public service organizations in the national as well as the international market. Our scientific and technical capabilities are further developed in co-operation with The Research Council of Norway and key customers. The results of our projects may take the form of reports, software, prototypes, and short courses. A proof of the confidence and appreciation our clients have for us is given by the fact that most of our new contracts are signed with previous customers.

Title	ITLED4240 Compendium Spring 2012: Open Source, Open Collaboration and Innovation
Editors	Wolfgang Leister, Nils Christophersen
Quality assurance	Trenton Schulz
Date	March 1, 2012
Publication number	DART/01/2012
Contributors	Wolfgang Leister, Nils Christophersen, and Arne-Kristian Groven; with contributions from Kirsten Haaland, Ruediger Glott, Anna Tannenberg, and Xavier Darbousset-Chong

Abstract

This document is the compendium for the course *ITLED 4240: Open source, open collaboration and innovation* at the Department of Informatics, University of Oslo, written for the spring semester of 2012. This multi-disciplinary compendium presents aspects of commons-based peer production, free and open source software, open licenses, open standards, and how to extract value from these phenomena in open innovation and business models.

Keywords	free software, open source, open collaboration, open innovation, open licenses, open standards
Target group	Students and Researchers
Availability	Open
Project	ITLED4240V12
Project number	
Research field	Commons-Based Peer Production
Number of pages	139

© 2010–2012 by Wolfgang Leister, Nils Christophersen, and contributors. This collection is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.



Contents

1	Introduction	9
2	Examples of Commons-Based Peer Production (CBPP) by Nils Christophersen and Wolfgang Leister	15
2.1	Free and Open Source Software	16
2.2	Wikipedia	17
2.3	OpenStreetMap	19
2.4	Genealogy	21
2.5	Volunteer Computing	23
2.6	Geocaching	24
2.7	Project Gutenberg	25
2.8	Open hardware	25
3	Characterisation, Management, and Production of Goods by Nils Christophersen	27
3.1	Characterisation of Goods	28
3.2	Traditional Production and Management of Goods	31
3.3	Production and management of goods through commons-regimes	33
3.4	Scientific communities	35
3.5	Commons-Based Peer Production	38
3.6	Open Innovation	42
4	Free and Open Source Software by Wolfgang Leister	47
4.1	FOSS Concepts	47
4.2	Historical and Societal Context	50
4.3	Software Licenses	53
4.4	FOSS Distributions	58
4.5	Development Process	60
4.6	Costs of FOSS	62
4.7	FOSS Business Models	63
4.8	Characteristics of FOSS	67
5	Quality Assessment of FOSS by Arne-Kristian Groven, Kirsten Haaland, Ruediger Glott, Anna Tannenbergh, and Xavier Darbousset-Chong	79
5.1	Software Quality Models	80

5.2	OpenBRR.	82
5.3	QualOSS	85
5.4	Experiences and Results from Assessments	88
5.5	Comparing the Assessment Methods	92
5.6	Concluding Remarks	94
6	Open Licensing.	99
	by Wolfgang Leister	
6.1	Creative Commons	100
6.2	Open Knowledge and Open Data	109
6.3	Governmental Licenses	117
6.4	Hardware Licenses	119
6.5	Equity-based Licenses.	121
6.6	Case Study: Licensing in OpenStreetMap	124
7	Open Standards	129
	by Wolfgang Leister	
7.1	Open Standards	130
7.2	Embrace, extend, and extinguish	133
7.3	Open Formats	134
7.4	Standardisation and the Public Sector	134
7.5	Patents and Standards	137
7.6	Case Study: Video Codecs in HTML5.	138

Preface

This is the compendium for the course *ITLED 4240: Open source, open collaboration and innovation*¹ at the department of informatics, University of Oslo. This document is written for the spring semester of 2012, and developed from the course *INF 5780*² with the same title. The main theme is the study of various communities participating in open collaboration on the Internet. This large and diverse field is still under rapid development with high degrees of creativity and innovation. It is multidisciplinary including the role of technology, legal issues, cultural aspects, business, markets, and public institutions.

A first version of a similar course was established in 2008. We thank Prodromos Tsiavos for his help in setting up the course initially with its emphasis on the concept Commons-Based Peer Production (CBPP). Students who participated previously in the courses have provided valuable contributions. We also thank Karthik Jayaraman, who is a doctoral researcher at the University of Oslo, for contributions to this course. For further improvements and updates we depend on close interactions with a community of students and others who are interested in this area. The authors thank Michel Bauwens for discussions and inspiring lectures. The authors are grateful to Trenton Schulz for valuable comments and proofreading of the manuscript.

In our own work, we prefer Free and Open Source Software (FOSS) when this offers good solutions. In that spirit, this compendium is edited using the typesetting system \LaTeX , the text editor Emacs, the Linux operating system, the mindmapping software vym, the diagram software dia, and several others.

The compendium is licenced under a Creative Commons Attribution - Non-Commercial - ShareAlike 3.0 licence (CC-BY-NC-SA)³. This is the same licence as used by, e.g., MIT for its Open courseware⁴. Briefly, this means that the content can be used freely by anyone for non-commercial purposes, provided attribution is given. In addition, any material based on or derived from this compendium must also be released under the same licence.

Oslo, March 1, 2012

Wolfgang Leister

Nils Christophersen

1. See <http://www.uio.no/studier/evu/kurs/matnat/informatikk/it-og-ledelse/ITLED4240/>; accessed February 18, 2012.

2. See <http://www.uio.no/studier/emner/matnat/ifi/INF5780/>; accessed July 25, 2011.

3. See <http://creativecommons.org/>; accessed July 25, 2011 and also the Norwegian Creative Commons <http://creativecommons.no/>; accessed July 25, 2011.

4. See <http://www.ocw.mit.edu/terms/>; accessed July 25, 2011.

1 Introduction



Everything that can be represented as digital files, can now, given sufficient bandwidth, be distributed and shared on the Internet. In addition, search engines make it possible to discover and retrieve material of interest from what is, effectively, an immense and chaotic library. A striking phenomenon – often denoted as Web 2.0 – is the degree to which many people have taken advantage of this potential by becoming participants and producers instead of being only consumers. The term *prosumer*, coined early by futurologist Alvin Toffler (1980), aptly describes this new mode of participation in contrast to consumers in the traditional *one-to-many* mass media era¹.

Web 2.0, characterised by user-generated content provided voluntarily and free of charge, has given individuals unprecedented possibilities to express themselves to a global audience and taken participation and collaboration between people to new levels. With the low transaction and coordination costs offered by web communities, people participate and collaborate in communities of all kinds, from socialising on Facebook to contributing to large and complex projects within Free and Open Source Software (FOSS).

Looking at the current picture, it is clear that this opening of communication, constant involvement, and interchange on a large scale have set in motion several disruptive phenomena and led to innovation that would have been unexpected from a conventional viewpoint, where the leading roles of formal organisations, experts, and professionals are taken for granted. New developments within many, if not most, areas of society are providing rich grounds for interdisciplinary studies and research. Many established paradigms in the social sciences relating to human collaboration and how innovation occurs have to be reconsidered in the light of what is taking place.

Internet-based communication and cooperation take place within the sphere of *social media*. Kaplan and Haenlein (2010) group this area into six broad and overlapping categories: (1) collaborative projects such as FOSS and *Wikipedia*, (2) blogs and microblogs such as *Twitter*, (3) content communities such as *Youtube*, (4) social networking sites such as *Facebook*, (5) virtual game worlds such as *World of Warcraft*, and (6) virtual social worlds such as *Second Life*. With our emphasis on open source, the focus of this course is the first category – collaborative projects.

Posing challenges from traditional points of view, much work has been devoted to studying this phenomenon on a more general and theoretical basis. One of the first to do so was

1. Note, however, that a so-called participation inequality typically holds for Internet communities – in relative terms most people still only view and do not contribute, cf. Chapter 2 and Chapter 3. However, with the massive number of people on-line, the absolute number of contributors is often huge.

the legal scholar Yochai Benkler (2002), who coined the term *Commons-Based Peer Production* (CBPP). This term relates to communities consisting of peers, i.e., people working together voluntarily without a formally imposed organisation or hierarchy, which produce a shared resource – a commons² – open to everyone free of charge. A commons may generally be defined as a resource or good shared by a group of people that may be subject to social dilemmas such as competition for use, free riding and over-exploitation (Ostrom, 1990). Its main original meanings relate to shared natural resources in traditional societies, e.g., grazing fields and fishing rights, and to communal meeting places, e.g., the Boston commons park and US town halls, but is now also applied to the wholly different domain of shared resources on the Internet.

This community-based approach originated with FOSS in the 1980s and has provided inspiration and been a kind of role model for much of the current activities. FOSS is radically different from so-called *proprietary software* provided by commercial vendors. There are thousands of FOSS projects and some of these (e.g., the GNU/Linux operating system and the Apache web server software) are large, innovative, and hugely significant within the software field. The FOSS source code is freely available for anyone to download, and people are encouraged to do just that. This contrasts sharply with commercial software, where the users are only provided with the computer-readable, binary code, and instructions on how to use it.

There is now a huge literature on FOSS, CBPP and related phenomena such as how established organisations and companies may harness the potential of Internet communities. Good repositories for scientific papers are the DBLP computer science bibliography³ and the Digital Library of the Commons at Indiana University⁴. Books with more comprehensive and accessible treatments include Howe (2008) about crowdsourcing, von Hippel (2005) about democratising innovation, Chesbrough et al. (2006) about open innovation, Surowiecki (2004) about “the wisdom of crowds”, Tapscott and Williams (2006) about “wikinomics”, Leadbeater (2008) about “we-think”, and Shirky (2010) about “cognitive surplus”. The *Foundation for P2P Alternatives* or the *p2pfoundation*⁵ is one site that promotes peer production and hosts a large amount of material and lists hundreds of CBPP projects within software, media, production of physical goods, and other areas.

In studying how Internet-based communities produce, create, and innovate, important questions relate to why volunteers from all over the world work jointly in this way, even if often only relatively few may know each other personally? Furthermore, how do such diverse groups self-organise to produce very complex products that are both free of charge and, in some cases, better and more innovative than similar commercial products, thus beating established businesses at their own game? Such questions again

2. “Commons” is an awkward word in English being the same in both plural and singular forms. The Norwegian term is “allmenning”; see <http://no.wikipedia.org/wiki/Allmenning> while the German term is “Allmende”; see <http://de.wikipedia.org/wiki/Allmende>.

3. See <http://www.informatik.uni-trier.de/~ley/db/>; accessed August 2, 2011.

4. See <http://ldlc.dlib.indiana.edu/dic/>; accessed August 2, 2011.

5. See http://p2pfoundation.net/The_Foundation_for_P2P_Alternatives; accessed August 2, 2011. Here, p2p stands for peer-to-peer.

pose puzzles from traditional management and economic standpoints where the underlying assumption is that goods and services of economic significance are either produced by companies in a market or provided by the government or other public institutions. CBPP, on the other hand, is part of the so-called *gift economy*⁶ where goods and services are provided voluntarily without expecting any immediate monetary or other type of compensation.

Organisation of this compendium

In Chapter 2, we briefly describe some of these projects to show typical characteristics and indicate the existing diversity and experimentation. In addition to FOSS, we show as examples projects like Wikipedia, OpenStreetMap, work within genealogy, so-called *geocaching*, and open source hardware.

Chapter 3 explores the phenomenon of CBPP in a broader context building in particular on work by Benkler (2002, 2007); Hess and Ostrom (2007); Ostrom (2005), and Weber (2004). CBPP is compared with the traditional ways goods and services are produced – in markets or by public institutions. Long-standing questions in established economics and political debates relate to the relative roles of the market versus the public sector. What should be left for the market to produce and what should be the responsibility of the of the public sector? With CBPP as a third and prominent mode of production as emphasised by Benkler, new questions arise as to how this mode will relate to the established domains. At present, we are in a transition period where the outcome is hard to judge. This is exemplified by the so-called Carr-Benkler wager (bet)⁷ set up in 2006. Benkler maintained that by 2011 most of the content on major Internet sites would be produced through CBPP, while the writer Nicholas Carr, on the other hand, disagreed arguing that CBPP by then would be absorbed by the traditional sectors, in the sense that most of the content would be generated by people paid from the traditional sectors. The wager has apparently not yet been resolved between the two, but what we see may be characterised by the term *disruptive innovation*⁸, where new opportunities open up and gradually displace and disrupt traditional ways of doing things in both the market and in public institutions. This is also related to the concept of *open innovation* which describes how companies may be able to draw on a community in order enhance innovation.

Further in Chapter 3, we explain the difference between traditional commons and knowledge commons. In this context, it is important to note that traditional commons, relating to natural resources, with seemingly good reasons have been considered fringe phenomena in the modern market-oriented economy. The concept was, in fact, almost discredited⁹ until Elinor Ostrom and coworkers showed that under certain conditions this mode of organisation can indeed be very effective for local communities to manage natural re-

6. See http://en.wikipedia.org/wiki/Gift_economy; accessed August 29, 2011.

7. http://en.wikipedia.org/wiki/Carr_Benkler_wager; accessed July 28, 2011.

8. http://en.wikipedia.org/wiki/Disruptive_innovation; accessed August 6, 2011.

9. The phrase “tragedy of the commons”, coined by Hardin (1968), and explained in Chapter 3 comes to mind.

sources¹⁰. It also turns out that CBPP communities have many similarities with scientific communities, working outside the market with their emphasis on open communication and peer review.

Intellectual Property Rights (IPR) is a hotly contested area, not least regarding patents and copyright which play crucial roles in the market economy.¹¹ Copyright and licences (i.e., specially designed forms of copyright terms) are briefly considered in Chapter 3 with more material on FOSS licenses in Section 4.3, and the Creative Commons and licenses for other content and goods in Chapter 6.

Looking at FOSS in Chapter 4, we take a non-technical approach meaning that skills in coding are not required for this course. Instead, we give a more thorough exposition of the history of FOSS, the particular ways FOSS communities are organised and work, the use of licences and main areas of applications. Business models for FOSS are part of an interesting area. Although the source code is freely available, there are several ways in which one can earn money based on FOSS. In Chapter 5, the issue of quality assessment of FOSS is treated in some depth. With the code being freely available without warranty, the question of quality assurance requires new approaches. Different approaches for the assessment and the used metrics are presented.

While licenses for FOSS already are discussed in Chapter 4, we look closer into open licensing for other content and goods in Chapter 6. The Creative Commons Licenses for content, the open licenses for data and databases, licenses for public sector data, hardware licenses as well as the more radical equity-based licenses are considered.

Open Standards are an important issue for FOSS and open innovation in general. Chapter 7 presents definitions of what open standards and open formats are, and discusses their roles. We also consider the role of open standards in the public sector.

As mentioned, this compendium is work in progress, and we are aware that some subjects need further elaboration. Important topics, such as normative issues, i.e., discussions about which values should be promoted by technological as well legal frameworks, are not yet treated. Despite this, we offer a fairly broad text book containing important areas in a world dominated by the Internet¹². Hopefully, our course is of interest to a wide range of students from disciplines beyond computer science. It is our experience that although students are proficient Internet users and participate in various online communities, many lack a deeper understanding of underlying central issues. Where we might have been too superficial or not clear enough, one can always explore the issues further on the *Net* itself.

10. Elinor Ostrom received the economics prize in the memory of Alfred Nobel in 2009 for her work.

11. Some people, for example Richard Stallman (2002) in the Free Software Foundation, discard the term IPR altogether. Since this is not property in a conventional sense but time-limited monopolies granted creators by society before the creation becomes freely available to all, Stallman argues against the use of “property”.

12. We are not aware of many similarly broad courses elsewhere. But one example of such a course is given at UC Berkeley; see <http://www.ischool.berkeley.edu/courses/290-cbpp>; accessed June 8, 2011.

References

- Yochai Benkler. Coase's Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal*, 112:369–446, 2002.
- Yochai Benkler. *The Wealth of Networks*. Yale University Press, October 2007. ISBN 0300125771, 9780300125771.
- Henry William Chesbrough, Wim Vanhaverbeke, and Joel West. *Open Innovation: The New Imperative for Creating And Profiting from Technology*. Oxford University Press, August 2006. ISBN 0199290725, 9780199290727.
- G. Hardin. The tragedy of the commons. *Science*, 162(3859):1243–48, 1968.
- Charlotte Hess and Elinor Ostrom. *Introduction: An Overview of the Knowledge Commons, In Understanding Knowledge as a Commons*. MIT Press, 2007. ISBN 0-262-08357-4.
- Jeff Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Business, first edition, August 2008. ISBN 0307396207.
- Andreas M. Kaplan and Michael Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59–68, 2010. ISSN 0007-6813.
- Charles Leadbeater. *We-think: Mass innovation, not mass production: The Power of Mass Creativity*. Profile Books, illustrated edition edition, February 2008. ISBN 1861978928.
- Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, 1990. ISBN 0521405998.
- Elinor Ostrom. *Understanding Institutional Diversity*. Princeton University Press, 2005. ISBN 0691122385.
- Clay Shirky. *Cognitive surplus, creativity and generosity in a connected age*. Penguin books, 2010. ISBN 978-1-846-14218-5.
- Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, Massachusetts, 2002.
- James Surowiecki. *The wisdom of crowds: why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations*. Doubleday, 2004. ISBN 9780385503860.
- Don Tapscott and Anthony D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover, first edition Tenth Printing edition, December 2006. ISBN 1591841380.
- Alvin Toffler. *The Third Wave*. Bantam Books, 1980. ISBN 0-517-32719-8.
- Eric von Hippel. *Democratizing Innovation*. MIT Press, 2005. ISBN 0262002744, 9780262002745.
- Steven Weber. *The Success of Open Source*. Harvard University Press, April 2004. ISBN 0674012925.

2 Examples of Commons-Based Peer Production (CBPP)



by Nils Christophersen and Wolfgang Leister

In the following sections, we highlight some CBPP projects to point out common characteristics and to indicate the existing diversity and creativity. Many of these projects are listed by the *Foundation for P2P Alternatives*, or the *p2pfoundation*¹ which hosts a large amount of material and lists hundreds of such projects in various stages, some successful and some not.

Given the large diversity, it is not surprising the quality and usefulness of many CBPP projects can be questioned. An enthusiastic founding group alone is no guarantee for a project to be successful. Other factors are crucial, such as internal organisation of the community, ideas, resources, support and funding, outside interest — and a good portion of luck. Some of the example projects we have chosen are highly successful and well known, while others are less so.

Before treating the examples, we give, as a reference, an informal list characterising CBPP projects, partly based on Benkler (2002). This will be further elaborated on in Chapter 3.

1. *Projects*. The project should be non-profit and driven by a community where the members volunteer in participating, all sharing a common goal or vision. Specially designed copyright licences regulate how the product can be used, and access be secured. Further, the project must be *modular* so that different participants can work in parallel. The possible set of tasks should require a range of different skills from the very simple to the more complex and time consuming.
2. *Peers*. The community members select for themselves what they want to do. A project with widely different tasks will provide incentives to a diverse community of people with a range of interests and motivation.

The members are recognised solely by their contributions to the project and not through external factors such as wealth, power, or formal education. An informal hierarchy, called a *meritocracy*, may develop on this basis where the senior members have earned their trust and authority through merit, i.e., through their contributions to the community. Such a hierarchy may develop naturally since Internet communities typically exhibit what is known as the *participation inequality*². This means that

1. See http://p2pfoundation.net/The_Foundation_for_P2P_Alternatives; accessed August 4, 2010.

2. See <http://en.wikipedia.org/wiki/90-9-1>; accessed August 4, 2010.

only a small fraction of the participants creates most of the content.

3. *Integration*. Integration is carried out through the Internet with low communication and organisation costs using tools such as wikis, version control systems, and mailing lists.

2.1 Free and Open Source Software

Free and Open Source Software (FOSS) is the most well-known CBPP phenomenon. We will treat FOSS, its history, culture, legal licences, opportunities and challenges in Chapters 4 and 5. Here, we present some introductory observations. FOSS, originating as a concept in the mid-1980s, is now a pervasive phenomenon. For example, one of the main Internet sites hosting FOSS projects, Sourceforge³, held, as of August 2011, more than 300 000 projects with more than two and a half millions registered users. Many of these projects are small, inactive, or not very professional. However, the numbers, taken together with those of other sites, indicate a significant global phenomenon. FOSS flagships include the GNU/Linux operating system, the Apache web server software, the MySQL database, the Firefox web browser, the OpenOffice office suite, and many others.

The basic idea of FOSS is that the computer source code should be available for inspection, modification, and redistribution for everybody. This contrasts with commercial and proprietary software where the users are only given access to the binary, computer readable code. With the source code open, anyone can read and adapt it. Successful FOSS projects attract a critical mass of active voluntary developers with different skills and interests; some provide inputs ranging from simple bug reports and translations to substantial patches, while others take care of design, testing, and integration. Disputes and disagreements are frequent. These are often resolved through mediation by senior members or through some kind of voting procedures. At any time, someone may legally take the source code and start a competing project creating a so-called fork, but interestingly enough this is not frequent.

FOSS includes a diverse set of communities crowding around two main camps: *free software* vs. *open source* – differing on ideological and philosophical grounds. The Free Software Foundation⁴ (FSF), established by Richard M. Stallman in 1985 as a reaction to increased commercialisation of software at the time, emphasises the philosophy of freedom (Stallman, 2002). Stallman worked at MIT in a scientific engineering environment where software used to be freely shared, and he has committed himself to this cause. In his opinion, no one should be able to use ownership of software to exclude others from it; every holder of the code should have the freedom to share it with others (if he or she wants so) but not necessarily at zero price. In Stallman's words, think of *free* as in *free speech*, not *free beer*. Thus you can, in principle, offer free software to the market for any price you choose, but in practise this is zero, which is the effective cost of Internet distribution. However, Stallman considers consulting as a different matter, which has been an

3. See Sourceforge.net; accessed August 4, 2010.

4. See <http://www.fsf.org>; accessed September 1, 2011.

important source of income for him over the years.

Copyright of software was legally established in the US in the late 1970s. To secure freedom and avoid that FSF software could be copyrighted in the conventional sense by anyone, Stallman invented the General Public Licence (GPL) in 1989, a clever *legal hack* popularly denoted as the *copyleft*. This is a legally binding software licence, based on standard copyright law for its enforcement, but turns the copyright idea on its head. The GPL *requires* that the source code for any distribution containing GPL software must be released in full⁵.

The *open source* camp, cf. Raymond (1999), on the other hand, focuses less on the ideals of freedom and sharing and more on the strengths of the peer-to-peer approach, i.e., CBPP, to develop high quality software. Some of the open source licences, such as the Berkeley Software Distribution (BSD) license will, in contrast to the GPL, allow code to be incorporated into proprietary software where only the binary code is subsequently released.

Raymond famously stated *Linus' law*, referring to the founder of Linux, Linus Torvalds:

Given enough eyeballs, all bugs are shallow.

With many users, some very competent, applying the software in different ways and environments, the probability of detecting errors and getting fixes is large. Linus' law implies that FOSS development needs to be done in a highly modular way. This contrasts sharply with the top-down and strongly managed way commercial software has often been developed. A classical account of the traditional approach and its problems is the book *The mythical man-month* by Fredrick Brooks (1995). He stated what is known as *Brooks's law*⁶, which essentially says that adding more developers to a project that is already late, will only lead to further delays. The reason is the increased administrative costs in management and communication in a traditional organization. Thus Brooks's law stands in contrast to Linus' law, emphasising the value of communal efforts over hierarchical ones. The idea of software development as a modular and communal Internet-based effort with the tools to carry it out, was a very important innovation emerging in the 1990s. As noted, this has been a major inspiration in other areas where CBPP is undertaken.

2.2 Wikipedia

The on-line encyclopedia Wikipedia is another extremely successful CBPP project. As of August 2011, the site had 282 language editions, where ten had more than 500 000 articles (roughly the number of topics in Encyclopedia Britannica which is considered a kind of "gold standard") and 38 had more than 100 000. The number of individual monthly users were about 400 millions, making it the fifth most popular site on the Internet. To foster cultural diversity, the Wikipedia project also offers editions in minority languages, artifi-

5. Copyright is often expressed as *all rights reserved* whereas copyleft can be characterised by *all rights reversed*.

6. See http://en.wikipedia.org/wiki/Brooks's_law; accessed September 3, 2011.

cial languages like Esperanto, and languages considered as dialects, such as alemanic or deitsch.

The Wikipedia project was started by Jimmy Wales and co-workers in 2001, more as an experiment inspired by FOSS, and without too high expectations. Before that, Wales had started the online Nupedia encyclopedia⁷ which was also open to voluntary contributions, but which relied on reviews by experts before publication. Nupedia never took off. However, to Wales' surprise, Wikipedia prospered spectacularly.

The content of Wikipedia is licenced under a so-called Creative Commons (CC) licence. Creative Commons⁸ is a foundation established by the legal scholar Lawrence Lessig in 2001. Its aim is to provide: *free licenses and other legal tools to mark creative work with the freedom the creator wants it to carry, so others can share, remix, use commercially, or any combination thereof.* The idea is to give the creator more choice above either standard copyright or the GPL.⁹

Briefly, all CC licences require that attribution is given to the creator. In addition, the creator can choose if the work should be made available on a commercial or non-commercial basis, whether to allow derivative works (e.g., remixes) based on the original, and whether to require that derived works, when allowed, should themselves be made available on the same terms (Share Alike or copyleft). Thus, CC offers six licences in total, since not all permutations of the options are meaningful. These licences have now become de-facto standards for content licensing beyond software, databases and hardware. Open licensing will be treated in detail in Chapter 6.

Wikipedia uses the *CC Attribution Share Alike*¹⁰ licence which is, in fact, close to the spirit of the GPL. This means that anyone can use Wikipedia content commercially provided they give attribution and cover what they then produce with the very same licence.

In 2003, Wales transferred the legal rights to Wikipedia to the Wikimedia foundation which now has nine other sister projects including the free media repository Wikimedia Commons. The scale of the voluntary effort is evident from the fact that as of August 2011, only about 75 people were employed by the Wikimedia foundation which is funded through donations.

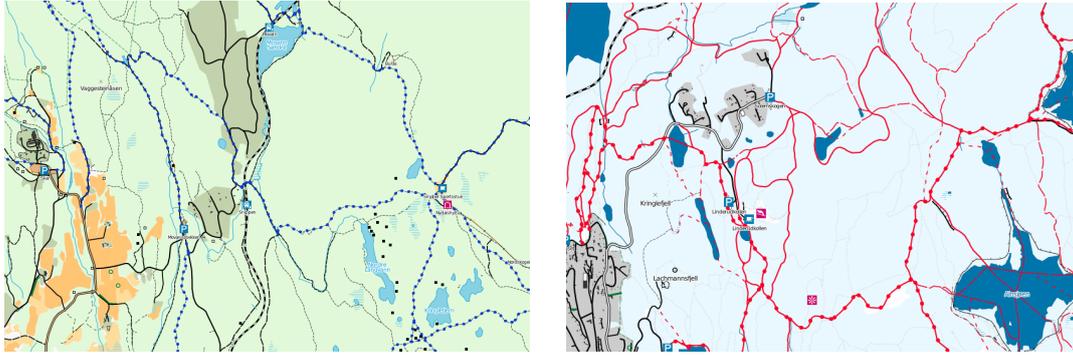
Wikipedia has several typical CBPP characteristics. While centrally based standards are required, work on different entries and topics is to a large extent modular, allowing people to concentrate on their area of interest. The volunteers take on different tasks from correcting language and improving indexing, to writing and maintaining larger articles. The contributions rather than the formal qualifications matter. For example, a professor

7. See <http://en.wikipedia.org/wiki/Nupedia>; accessed September 3, 2011.

8. See <http://creativecommons.org/>; accessed August 4, 2010.

9. Note that the GPL is designed for software, thus often not being suitable for other use. The GNU Document License can be used for all types of documents. However, the CC is designed for other types of media as well.

10. **CC-BY-SA.** See http://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License; accessed August 4, 2010.



Images © OpenStreetMap Contributors CC BY-SA

Figure 2.1. Examples of map renderings in OpenStreetMap. Hiking trails (left) and cross country skiing trails (right) in areas of the Lillomarka near Oslo.

in one area will have little merit within his topic in Wikipedia until he or she has made contributions deemed significant enough by the community. Based on such merit, a formal hierarchy has developed from editors, sysops, and bureaucrats, to stewards with increasing responsibilities and rights for such tasks as mediating disputes, deleting or locking controversial articles for further edits, and excluding community members.

Generally, Wikipedia holds good quality standards. For example, an investigation by the reputed journal *Nature* (Giles, 2005) found about the same number of mistakes in Wikipedia as in *Encyclopedia Britannica*¹¹. However, whereas the mistakes in Wikipedia were corrected almost instantaneously, Britannica has a slow review process to correct errors.

2.3 OpenStreetMap

OpenStreetMap (OSM)¹² works with creating a data-base of geospatial data, i.e., all kinds of facts that can be used to derive maps for diverse purposes. Currently, most OSM data are licenced under the Creative Commons **CC-BY-SA** licence as Wikipedia. Some data are in the public domain, i.e., not covered by legal restriction.

Besides the ordinary map, maps for cycling, hiking, skiing, nautical purposes, public transport, or maps for use in a GPS device¹³ can be derived from the OSM data. We show examples of map renderings in Figure 2.1. Also information brochures for touristic purposes, such as town guides have been created from the data.¹⁴ The OSM project also hosts several tools for mapping, rendering, and processing of these geospatial data.

11. See http://en.wikinews.org/wiki/Wikipedia_and_Britannica_about_as_accurate_in_science_entries,_reports_Nature; accessed August 4, 2010, http://en.wikipedia.org/wiki/Wikipedia:External_peer_review/Nature_December_2005; accessed August 4, 2010 and http://blogs.nature.com/wp/nascent/supplementary_information.pdf; accessed September 3, 2011. Note also that Britannica refuted some of the findings of this study; see http://corporate.britannica.com/britannica_nature_response.pdf; accessed September 3, 2011.

12. See <http://www.openstreetmap.org/>; accessed August 4, 2010.

13. See www.frikart.no; accessed August 4, 2010.

14. See code.google.com/p/townguide/; accessed April 18, 2011.

Geographic data from OpenStreetMap can be re-used for any purpose, while this is not the case with interactive map services such as Google maps¹⁵ While the maps are freely available for viewing, and overlays for other third-party data, such as tracks from a GPS receiver can be rendered and applied, the geodata itself are not available. In contrast, OpenStreetMap allows the geodata to be extracted, processed with other third-party data, and visualised as long as the licensing terms are followed.¹⁶

OSM was founded in July 2004 by Steve Coast. In April 2006, a foundation was established to encourage the growth, development and distribution of free geospatial data and provide geospatial data for anybody to use and share. Besides other milestones it is worth mentioning that in December 2006, Yahoo confirmed that OpenStreetMap could use its aerial photography as a backdrop for map production.¹⁷ At the end of 2010, *bing*, the search service operated by Microsoft, allowed aerial images from their service to be used at backdrop images.¹⁸ The use of aerial photography eases the collection of data, so that currently both images from Landsat and from Yahoo can be used.

By August 2008, shortly after the second *The State of the Map conference* was held, there were over 50,000 registered contributors; by March 2009 there were 100,000; by the end of 2009 nearly 200,000; in autumn 2010 nearly 300,000 contributors; and by August 2011 over 450,000.¹⁹

As found in other CBPP communities, a minority of the registered users contribute the majority of the content: in March 2008 approximately 10% of the registered user base were contributing to the map data each month. New numbers in summer 2010 show that now only ca. 5% are contributing to the map each month.²⁰

Many FOSS projects support OSM. The OSM project consists of a database running on a number of servers, a database format, a wiki, a conference series, map rendering software, and tools to import, maintain, retrieve, and present these data. The project has connections to separate FOSS projects, like *gpsbabel*²¹. Products for mapping, i.e., editing the data, include *Potlatch* (a build-in editor in the OSM web interface), *JOSM*, and *Merkaartor*. For rendering the maps alternatives such as *Mapnik*, *Osmarender*, and *Maperitive* are available.

15. See maps.google.com; accessed September 3, 2011.

16. For example, one could extract all roads in one area and visualise traffic patterns from a statistics database with varying thicknesses of these roads. This would be not possible with, e.g., Google maps, since the underlying geodata are not available for this purpose.

17. See <http://wiki.openstreetmap.org/wiki/Yahoo>; accessed August 4, 2010.

18. Note, that the use of bing maps for this purpose is not allowed. bing has announced that they create services based on OSM overlays. See wiki.openstreetmap.org/wiki/Bing; accessed August 30, 2011 and links pointing from there.

19. The data are retrieved from http://www.openstreetmap.org/stats/data_stats.html; accessed August 30, 2011. See also the fact sheet at http://community.cloudmade.com/blog/wp-content/uploads/2010/01/100106-OSM_Facts.pdf; accessed August 4, 2010.

20. The numbers are retrieved from <http://wiki.openstreetmap.org/wiki/Statistics>; accessed August 4, 2010.

21. See www.gpsbabel.org; accessed August 4, 2010. Note that *gpsbabel* is under the GPL, which implies that it cannot be used as a library together with proprietary software.

OSM is connected to related sister projects, like *Free the Postcode* (relating a geo-position to a post code), *Mapstraction* (making it possible to switch map in a browser interface), and *OpenStreetPhoto*, which seems to have split into *OpenStreetView* (geo-located photos) and *OpenAerialMap* (a not so successful attempt to exchange aerial imagery).

The *mappers*, as the contributors to the OSM community are called, collect geospatial evidence from their GPS receivers using tracks and marked waypoints, from knowledge, or from geo-located aerial imagery. Tracks and other data are uploaded using an editor, and labelled with the correct tags. To use the data, these need to be rendered, either customised on the user's PC, or using the web-interface.

The earthquake that struck Port-au-Prince in Haiti, January 12, 2010, started an intense activity to create a map of this area from aerial imagery and GPS traces on the ground, in order to help the first responders to the catastrophe, and to help humanitarian aid organisations. Within a short time, OSM hosted the only map of this area, that was precise enough for these purposes. The map also includes the many camps that were built for shelter after the quake. This example shows that CBPP principles in times of crisis are a very effective means. Therefore a separate web site, *CrisisCommons*²² has been created. There have also been efforts to use the concept of *crowdsourcing* for social activism and public accountability through *activist mapping*. One example is the non-profit company *Ushahidi*²³ which started by creating a witness-site after the heavily disputed election in Kenya in 2007.

The service FixMyStreet and its Norwegian counterpart FiksGataMi²⁴ build on data from OpenStreetMap. These services give citizens the opportunity to report problems with the infrastructure to their local council. The service automatically identifies the right authority from the description and the geo-location. Both services are interesting examples where CBPP is used to create a community that addresses real problems in the participants' lives, using artefacts from other CBPP projects, such as FOSS and geodata from OpenStreetMap.

2.4 Genealogy

The case of genealogy is interesting since it is said to be one of the largest activities on the Internet. Genealogy, the study of families and the tracing of their lineages and history²⁵, is a field that still struggles to embrace CBPP, even though most of the necessary pre-conditions seem to be in place. Besides scientists in history, there are many hobbyist genealogists who perform research for their own relatives with a high degree of volunteerism. While this field always has been important for the noblemen, it was embraced

22. See <http://crisiscommons.org/>; accessed August 4, 2010.

23. See www.ushahidi.com; accessed August 4, 2010 and en.wikipedia.org/wiki/Ushahidi; accessed August 4, 2010.

24. See www.fixmystreet.com; accessed September 3, 2011 and www.fiksgatami.no; accessed September 3, 2011.

25. See <http://en.wikipedia.org/wiki/Genealogy>; accessed August 4, 2010.

by larger parts of the people in the nineteenth century.²⁶ However, it seems that the CBPP principles are not very predominant, even though genealogists now use the Internet to perform their research. There are some examples, though.

Two issues are essential for genealogy: the genealogical data; and the tools to produce, process, and render these data, e.g., presenting, adding to the data base. Many of the original data sources for genealogy are centralised registers, driven by the command structure according to Benkler's classification. For Norway, the *Digitalarkivet*²⁷ maintained by the *Arkivverket* is one of these sources. There are also connections to projects to make these data available, such as the *historisk personregister*.

The Church of Jesus Christ of Latter-day Saints (LDS)²⁸ has engaged in large-scale micro-filming of records of genealogical value for their religious purposes, making these available through their Family History Library. The LDS also have developed the GEDCOM format for interchange of genealogical data²⁹ which has evolved to a de-facto standard. Due to the size of their library, the LDS are a predominant factor in genealogy. However, since the LDS have a hierarchical structure, they are not considered to be a driver in CBPP, but follow the command structure.

Lately, there are initiatives that can be considered CBPP, such as *WeRelate*³⁰, sponsored by the Foundation for On-Line Genealogy (FOLG), who collect facts and a genealogical database, predominantly the American area, e.g., providing lists of immigrants to America. For Norway, the *Lokalhistoriewiki*³¹ is a collection of genealogical data by the *Norsk lokalhistorisk institutt*.

The tools used in genealogy are mostly commercial software, or shareware developed by enthusiasts who market their program using the shareware mechanism.³² Among the FOSS genealogy tools we find *Gramps*, a project that was started in 2001 by Don Allingham, and released first in 2004 in a stable version³³. *Gramps* is an acronym for *Genealogical Research and Analysis Management Programming System*, and is licensed under the *GNU GPL, Version 2*. It is programmed in the programming language Python, has its own data base format (*Gramps XML*), and is capable of importing and exporting to file formats, such as GEDCOM.

26. Note that genealogy not always has been used for the good, since it has been used to prove whether a person belongs to a certain dependency, with the background of using this information for misguided political purposes.

27. See www.digitalarkivet.no; accessed Aug 10, 2010.

28. See http://en.wikipedia.org/wiki/The_Church_of_Jesus_Christ_of_Latter-day_Saints; accessed August 4, 2010.

29. See <http://en.wikipedia.org/wiki/GEDCOM>; accessed August 4, 2010.

30. See www.werelate.org; accessed August 10, 2010.

31. See www.lokalhistoriewiki.no; accessed August 10, 2010.

32. For a definition of *shareware*, see Section 4.1.

33. See gramps-project.org; accessed August 4, 2010.

2.5 Volunteer Computing

Volunteer computing³⁴ is a type of distributed computing in which computer owners donate their computing resources (such as processing power and storage) to one or more projects that benefit from these resources. Around the volunteer computing platform, and around the single projects there are communities whose members contribute with computing resources, and where the members' merits are visible through a ranking based on how much they contributed.

The Berkeley Open Infrastructure for Network Computing (BOINC)³⁵ is a middleware infrastructure developed at the Space Sciences Laboratory at the University of California, Berkeley led by David Anderson. BOINC uses the unused CPU and GPU cycles on a computer to do scientific computing. The BOINC software is released under the LGPL. BOINC consists of a server system and client software that communicate with each other to distribute, process, and return work units. The database including the computing chunks and the results is usually hosted by the project that benefits from the results.

BOINC was originally developed to manage the SETI@home project, replacing the original SETI client which was not designed with an adequate level of security, leading to cheating attempts on the "credits" and falsified work. The BOINC project started in February 2002 with Predictor@home as the first project in June 2004.

Besides SETI@home, examples of projects using the BOINC infrastructure include projects from biology, medicine, earth science, mathematics, physics, astronomy, fine arts, games, and others³⁶. Note that some of these projects are not CBPP projects, but scientific projects initiated by the scientific community. In several cases, the software used may not be released as FOSS.

For example, Folding@home is a volunteer computing project on protein folding and other molecular dynamics, launched by the Pande Group within Stanford University. The project source code is not available to the public, citing security and integrity concerns. A development version of Folding@home on BOINC framework remained unreleased.

Note that CBPP projects often are highly interconnected, and they often draw use from each other. The volunteer computing platform BOINC was developed from the infrastructure of the SETI@home project while BOINC now is used by a multitude of projects that might compete with SETI@home for the computing resources of the community participants.

There are energy consumption concerns looking at the energy balance between using dedicated computing resources or volunteer computing. Running a project on a modern personal computer will increase power consumption utilising the CPU often to a very high degree, and thus preventing the computer to go into the power-save mode. While

34. See http://en.wikipedia.org/wiki/Volunteer_computing; accessed August 7, 2010.

35. See <http://boinc.berkeley.edu/>; accessed August 7, 2010 and http://en.wikipedia.org/wiki/Berkeley_Open_Infrastructure_for_Network_Computing; accessed August 7, 2010.

36. See http://en.wikipedia.org/wiki/List_of_distributed_computing_projects; accessed August 7, 2010.

some may argue that the excess heat generated will contribute to house warming, in warmer regions there will be additional energy needed to cool the rooms. The energy-balance calculation requires a closer look.³⁷

Some types of crowdsourcing³⁸ are an interesting variant of volunteer computing, where human computing capacity is used to solve problems.³⁹ The Foldit project⁴⁰ aims at the area of molecular biology (Khatib et al., 2011); EteRNA aims to create synthetic RNA designs (Markoff, 2011); and Phylo wants to solve genetic sequences alignments⁴¹.

2.6 Geocaching

Leisure activities and games are an industry. Outdoor leisure activities using geographic data have become popular after the GPS technology has been available to the masses. One particular outdoor treasure-hunting game is *geocaching*⁴² where participants hide and seek containers called *geocaches*, *caches* for short. A typical cache is a small waterproof container containing a logbook and a “treasure”, e.g., trinkets of little value.

The caches are owned by members of the community, and announced on the website geocaching.com including coordinates, descriptions, hints and puzzles. Community members need a membership at the geocaching.com, with the possibility of a premium subscription. Cache owners are responsible to maintain their caches according to community rules. The community displays a ranking list, giving merits to the community members with numbers of caches found, maintained, etc. The web site is hosted by the company *Groundspeak, Inc.*, who also are involved in other games, such as *Wherigo*⁴³ and *Waymarking*⁴⁴ (marking interesting locations on a web site).

Wherigo is a mixture between geocaching and an adventure game where people can develop self-enclosed story files (the so-called “cartridges” that are installed on a GPS-enabled unit along with a player program. Reaching certain positions trigger events in the game, and completing an adventure can require reaching different locations and solving puzzles.

For all these games the software platform is proprietary, while the content is community based. The software is pre-installed on some devices, such as certain GARMIN GPS units. There is also the unofficial Openwig player for Java ME enabled devices. Geocaching has many aspects of CBPP with the geocaches, and the geographical information as the commons, using the tools provided by Groundspeak. Note that FOSS does not play any role here.

37. See <http://en.wikipedia.org/wiki/Folding@Home>; accessed August 7, 2010.

38. See en.wikipedia.org/crowdsourcing; accessed February 25, 2012.

39. Our thanks go to Erlend Vestad, Gunn Kristin Breien Johansen, and Helena Zafira Pedersen for making us aware of this phenomenon in their course assignment (Vestad, Johansen, and Pedersen, 2011).

40. See fold.it; accessed October 7, 2011.

41. See <http://phylo.cs.mcgill.ca/eng/>; accessed March 5, 2012 and en.wikipedia.org/wiki/Multiple_sequence_alignment; accessed March 5, 2012.

42. See www.geocaching.com; accessed August 7, 2010.

43. See <http://en.wikipedia.org/wiki/Wherigo>; accessed August 7, 2010.

44. See <http://en.wikipedia.org/wiki/Waymarking>; accessed August 7, 2010.

2.7 Project Gutenberg

Project Gutenberg (PG)⁴⁵ is probably the earliest project that can be considered as CBPP in our sense. It was founded by Michael Hart in 1971. The goal is “to provide as many e-books in as many formats as possible for the entire world to read in as many languages as possible.”

In 1971, the Internet was in its infancy connecting only mainframe computers at research institutions. Hart started by typing the United States Declaration of Independence by hand, and subsequently digitised over 300 books in this way. At that time, this was an unusual activity with a limited audience. Since then, the PG community has attracted volunteers carrying out activities such as digitising (by hand until 1989), proofreading, and performing administrative tasks.

The participants can select books to include on their own. In some cases, the project has permission to digitise books which are still under copyright, but the majority are in the public domain. Since copyright currently expires 70 years after the death of the author, only books whose authors died before 1941 can be freely included.

In the Project Gutenberg emphasis is put on accessibility and long term storage in open formats. Presently, the repository holds over 36 000 books in many languages with new releases steadily increasing.

Another site for accessing books is *Google books*⁴⁶ which now contains over 11 million books. The majority is still covered by copyright and Google makes only available “snippets” of such books. Note, however, that Google books cannot be considered as CBPP.

2.8 Open hardware

Open hardware or *open source hardware* is modelled on the basis of FOSS. The idea is to make the design of physical objects and manufacturing procedures freely available on the same terms as FOSS⁴⁷. As for FOSS, the creativity and commitment are extraordinary, showing a wealth of both serious and not so serious activities. For example, a visit to the Maker Faire will demonstrate this⁴⁸.

Open hardware has a lot of potential. A good example is the *RepRap*⁴⁹ 3D printer which is in the category of “desktop manufacturing”, allowing distributed production of physical objects at affordable prices. This has the potential to take Internet-based production and innovation to new areas. *RepRap* stands for “replicating rapid prototyper”, and can even print many of its own components. 3D printing is a form of additive manufacturing technology where a three dimensional object is created by successive layers of material.⁵⁰

45. See <http://www.gutenberg.org>; accessed August 24, 2011.

46. See <http://books.google.com/>; accessed Aug. 26, 2010.

47. See <http://www.openhardwaresummit.org/>; accessed August 1, 2011. Open hardware licences are treated in Section 6.4.

48. See <http://makerfaire.com/>; accessed August 1, 2011.

49. See http://reprap.org/wiki/Main_Page; accessed August 24, 2011.

50. Think about ink jet printing taken to 3D.

The *RepRap* project was started by Adrian Bowyer at Bath University, and now has participants and related activities in several countries. The goal of the project is to give individuals everywhere access to equipment that can produce artifacts for everyday life at a low cost. Today, commercial low-end 3D printers will cost from 10 000 USD. The parts for a RepRap, on the other hand, are available for about 400 USD. The RepRap design and the software are licenced under GPL.

References

- Yochai Benkler. Coase's Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal*, 112:369–446, 2002.
- Fredrick Brooks. *The Mythical Man Month and other essays on software engineering*. Addison-Wesley, 1995. ISBN 0785342835953.
- Jim Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901, December 2005. ISSN 0028-0836. doi: 10.1038/438900a.
- Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, Mariusz Jaskolski, and David Baker. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural & Molecular Biology*, advance online publication, September 2011. ISSN 1545-9993. doi: 10.1038/nsmb.2119.
- John Markoff. RNA game lets players help find a biological prize. *The New York Times*, page D4, January 10 2011. URL http://www.nytimes.com/2011/01/11/science/11rna.html?_r=3.
- Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, California, 1999.
- Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, Massachusetts, 2002.
- Erlend Vestad, Gunn Kristin Breien Johansen, and Helena Zafira Pedersen. Crowdsourcing science through games. course assignment, INF5780, Institute for Informatics, University of Oslo, 2011.

3 Characterisation, Management, and Production of Goods



by Nils Christophersen

Markets and public institutions have been the dominant sectors managing and producing *goods* in a modern society. Here we use the term *goods* in the broad economic sense including tangible man-made physical objects and natural resources as well as intangible intellectual goods, e.g., scientific knowledge, literature, music, and software. We also include in our definition what is often denoted as services, e.g., teaching, health care, and financial services.

Following Benkler (2002, 2007), we will treat CBPP as a third mode of production and consider its relationships to the traditional sectors. As noted in the Introduction, the current situation is not clear-cut and may be characterised by a period of *disruptive innovation*. On the one hand, CBPP creates tension and causes disruption, and, on the other hand, the innovation occurring creates new opportunities also for companies and public institutions.

The determining factors in economic production have been financial capital, technology, infrastructure, and human resources. In what is often denoted as the *information and knowledge economy*, the role of human motivation, creativity, and talent have become more important. With the advent of the Internet and the subsequent low technological and capital thresholds for cooperation and communication, the importance of these human factors are taken to new levels.

Public institutions are organised as hierarchies with formal lines of authority. In a market, companies are also hierarchies while the drive towards economic gains is central to motivation. CBPP communities, on the other hand, work in very different ways having no formal hierarchies and pursue no direct economic gains. Such communities are not new; non-profit organisations such as charities, foundations and Non Governmental Organisations (NGOs) have similar characteristics. However, contrary to CBPP, traditional non-profit organisations cannot generally be considered important factors in a society for managing and producing goods.

To study the potential of CBPP as well as its limitations and the types of goods that may be produced in this way, we need some background on how goods are characterised in general, and how they are produced and managed.

3.1 Characterisation of Goods

In economics, many questions relate to how firms work and compete in a market and whether goods should be produced or managed by public institutions or private enterprises. In such analyses, the two concepts of goods being *rival* vs. *non-rival* and *excludable* vs. *non-excludable* play important roles. Simply stated, this is because rivalness is related to scarcity or abundance, while excludability is related to property and whether or not access is granted and, if so, on which terms. These concepts are also central in discussing the role of CBPP and will be introduced in the following.

A good is *rival* if its use or consumption by one person limits or affects its use or consumption by another person; otherwise it is *non-rival*. Note that this is not an absolute property; in practise it is often more useful to talk about degrees of rivalness. Single personal physical objects such as clothes and books are always rival – they can only be used by one person at a time. Regarding natural or man-made resources, rivalness is related to scarcity. For example, an originally abundant and therefore non-rival water resource will become rival as water becomes scarce for some reason.

Technology may strongly influence rivalness. An example is telephony: fixed line phones were previously rationed by telephone companies in several countries, and thus rival. These are now replaced by abundant mobile phones making the acquisition and use of phones effectively non-rival almost everywhere. As an opposite example, where technology makes a previously abundant resource rival, consider the increased CO₂ levels in the atmosphere due to industrial development. Under a quota system limiting global emissions, national CO₂ emissions become rival since an increased quota for one country will imply reductions somewhere else.

Intellectual goods such as information, knowledge, and culture are considered inherently non-rival, since one person's use will not subtract from that of another person. While a book as a physical object is rival, the novel as such is non-rival. The book is said to be an *expression* of the novel in the same manner as a CD is an expression of a piece of music. The non-rivalness of intellectual goods is well expressed in a famous quote by the third president of the United States, Thomas Jefferson, who strongly influenced the country's copyright and patent laws. He wrote to a friend in 1813:

*He who receives an idea from me, receives instruction himself without lessening mine; as he who lights his taper [candle] at mine, receives light without darkening me.*¹

Goods may also have the property of being *anti-rival* (Weber, 2004). This means that as more people start using the good, the value increases for all. Such a phenomenon is also called a positive network effect and has been popularised through *Metcalfe's law* which states that the value roughly increases as the square of the size of the community². The Internet and other networks are examples of this. As more people participate, even more

1. <http://odur.let.rug.nl/~usa/P/tj3/writings/brf/jef1220.htm>; accessed August 26, 2010.

2. http://en.wikipedia.org/wiki/Metcalfe's_Law; accessed August 6, 2011. Metcalfe's Law is related to the fact that the number of unique connections in a network of n persons can be expressed as $n(n-1)/2$, which is proportional to n^2 for large n , i.e. $O(n^2)$

	Non-rival	Rival
Non-excludable	<i>Public goods</i> ; e.g., funded through public institutions including scientific knowledge, non-toll roads, policing, defence; goods provided by CBPP communities ^a .	<i>Common-pool resources</i> ; e.g., water for irrigation, atmospheric CO ₂ levels related to climate change with no proper enforcement of quotas.
Excludable	<i>Toll or subscription goods</i> ; e.g., toll roads, newspaper subscriptions, intellectual goods under conventional IPR. ^b	<i>Private goods</i> ; e.g., personal items such as clothes, CDs, and books; private property.

a. There is a catch here. These goods are typically protected by copyright. But then by a special licence securing re-use and re-mix such as GPL or Creative Commons.

b. Such as standard copyright – all rights reserved

Table 3.1. The four types of goods characterised according to whether they are rival/non-rival and excludable/non-excludable. Adapted from Hess and Ostrom (2007).

people will tend to join the community, in turn even further increasing the usefulness or value of participating.

A good is *excludable* or *non-excludable*, if someone has the right to regulate access to it or not. Issues of excludability are independent of rivalness. The right to grant access, either for free or for a price, or deny it altogether, is the essence of ownership. Property laws are, therefore, central to excludability. Intellectual goods are subject to Intellectual Property Rights (IPR). Note that these do not imply ownership in a conventional sense but give instead a time-limited monopoly, as discussed below.

In addition to legal rights, measures of exclusion include physical barriers and secrecy. As with rivalness, the *degree* of excludability is often of most interest and this depends on how effectively the available measures may be enforced in practise. Technological changes may strongly influence excludability. An obvious example is the role of the Internet and PC in allowing large-scale file sharing of copyrighted material. Earlier, it was hard for most people to copy CDs and books so this was a technical barrier in addition to the legal one, i.e., the IPR. With advent of the Internet this barrier disappeared and instead copy protection measures such as Digital Rights Management (DRM) were developed³.

Considering these concepts together leads to a broad division of goods into four groups as shown in Table 3.1. Rival goods are called *common-pool resources* if they are also non-excludable and thus vulnerable to competition leading to over-use. They are called *private goods* (typically owned privately) if they are excludable. Non-rival goods on the other hand, are aptly called *public goods* if they are non-excludable and thus available

3. http://en.wikipedia.org/wiki/Digital_rights_management; accessed August 10, 2011.

to anyone. They are called *toll and subscription goods* if they are excludable.

We will consider the above division in more detail below and how it relates to production in the market, in the public sector, and CBPP. But given the importance of IPR as a measure of exclusion in our context, we give a brief treatment of some of the legal measures and issues in this complicated and contested field. For physical objects, land, and natural resources, the concept of property is generally well understood and established. However, this is not necessarily the case for goods covered by IPR.

3.1.1 Excludability through IPR

Starting with the basics, two main instruments of IPR laws are *patents* and *copyright*; others include trademarks, industrial design rights and trade secrets. Patents and copyright have different histories and were developed for different purposes. Patents have been issued by authorities for centuries and regulate the use of inventions and ideas for commercial use. Copyright, on the other hand, does not cover ideas as such, but applies to the “expression” of works (copying, distribution, and adaption), such as printed matter, drama, sound recordings, movies, and computer programs. Copyright came into use through the invention of the printing press, and the first modern copyright law was the British Statute of Anne in 1709.

Both patents and copyright allow goods to be made excludable only for a limited period, thus creating a monopoly for a certain time. Therefore IPR does not imply “property” in the conventional sense. For this reason, Richard Stallman (2002), for example, in the Free Software Foundation discards the term IPR altogether. In most jurisdictions (geographical area covered by the same IPR regime), the time limits are now 20 years for patents and 70 years after the death of the creator (or last surviving creator) for copyright⁴. Thereafter, the good is not covered by IPR and passes into what is called the *public domain* available to all. To obtain protection, a patent must be applied for and the application must contain a description of the invention which, if the patent is granted, is made public. In return, the inventor gets the exclusive but time-limited commercial rights. Copyright, on the other hand, applies automatically and need no application to be valid.

The reason for the time limits rests on what we may call a *fundamental trade-off* where the law tries to strike a balance between the interests of the creator on the one hand, and the society at large, on the other hand. By granting a monopoly, society gives the creator the possibility to reclaim the investments and development costs, and make a profit in the market. This, in turn, is meant to stimulate further work and innovation by the creator, possibly allowing more risk-taking. But by granting only a time-limited monopoly, the invention or work will eventually become non-excludable and a public good, available for everyone to use and build on, which will promote general progress and social welfare.

It seems reasonable to give creators certain rights, or at least, no one will argue that creators of intellectual goods shall *not* be able to earn a living from their work. The main

4. Here the “creator” is the person(s) or the legal entity (e.g., company) holding the legal rights. These rights may be transferred or sold.

controversies are about where to strike the balance and how the rights should be implemented and exercised in a given jurisdiction. Concerns have been raised by several legal scholars, e.g., Benkler (2002), Lessig (2005), Heller (2008), and Boyle (2008) that the balance has been tipping too much in favour of the creators which in practise often means commercial actors. For example, allowing copyright to last 70 years after the death of authors seems overly long. A shorter period could hardly lessen the motivation of authors to pursue writing and publishers to publish their work. Today, only works by authors dead before 1941 are public goods.

When Britain introduced its copyright law in 1709, the duration was 14 years and could be renewed for another 14 years if the author was still alive after the first term expired. The US used the same time limits when introducing its federal Copyright Act of 1790. The strengthening of copyright laws over the years have largely been promoted by commercial interests. A notable and more recent case is the Digital Millennium Copyright Act (DMCA)⁵ which was passed in the US in 1998. Among other things, it strengthened penalties for copyright infringement on the Internet. Thus, as copying and distribution became simple in recent years reducing the role of commercial actors, copyright laws and their enforcement were strengthened. A debate in *The Economist*⁶ in 2009 gives insights into the arguments framed on both sides.

Regarding patents, the dispute is not so much over the issue of the duration as over over-patenting. Over-patenting includes protecting ideas with little *innovation height* (e.g., Amazon's patented *one-click* buy option) and creating of *patent thickets* (dense webs of overlapping legal rights). Questions of fairness have also been raised such as charging equally high prices for pharmaceuticals in developed as well as developing countries.

Heller (2008) introduced the term "gridlock" economy to describe a situation where excessive legal rights block access to intellectual goods thus impeding and not promoting progress and social welfare. However, as a result of the success of CBPP and in particular FOSS, an opposing trend emphasising even the commercial value of openness and collaboration has emerged. For example, companies in a market may find FOSS useful as part of their products to increase quality and also as a tactic to undermine similar proprietary software from competing companies. See, e.g., Lerner and Schankerman (2010). We stress again that CBPP products are typical under copyright law, but covered by special licences such as the GPL and Creative Commons licences. These allow the creators to enforce the specific types of open access and use they will grant.

3.2 Traditional Production and Management of Goods

Important questions in politics and traditional economics at the heart of many issues and controversies, are how firms should work and compete in a market, and whether goods should be produced or managed by public institutions or by private companies.

In a competitive world, excludability through clear property rights is a prerequisite for

5. <http://en.wikipedia.org/wiki/DMCA>; accessed August 10, 2011.

6. <http://www.economist.com/debate/debates/archive/page:13>; accessed August 10, 2011.

the market economy to work. Without such rights, companies cannot negotiate and trade. Referring to Table 3.1, the market can therefore only manage and provide goods in the two lower quadrants – toll or subscription goods or private goods. The role of the public sector is to regulate the market, and to provide goods that the market cannot readily supply, or that society determines should be non-excludable and non-rival so that everyone benefits. A main domain of the public sector is thus the upper left quadrant aptly named *public goods*. However, society may also provide goods at a price paid up-front such as water and electricity, which are then excludable subscription goods. The common pool resources in the upper right quadrant play no large role in the market vs. public dichotomy.

In the traditional sectors, economic gains, formal contracts, and hierarchical organisations with clear lines of authority provide incentives and regulate behaviour. The importance of reciprocity and trust between actors are, of course, recognised but are not assumed critical for the market or the hierarchy to work. An important advantage of these standard forms of organisations is that markets and hierarchies scale; i.e., since they are not based on personal relationships, they function even as the number of people involved grows by orders of magnitude. However, with sufficiently low communication and transaction costs, CBPP demonstrates that humans can still carry out large collaborative and economically significant projects without hierarchy and monetary incentives.

The underlying view of humans in the traditional spheres of economics is one of rational and self-interested beings narrowly focussed on their subjective ends. This view has been characterised by the term “*homo economicus*”⁷. In line with this view, it is assumed that people acting rationally will not contribute voluntarily to a common task if their contribution is only insignificant to the whole effort. This was stated already by Olson (1965) in his influential book *The Logic of Collective Action: Public Goods and the Theory of Groups* that almost became classic in parts of the social sciences. He further theorised that “only a separate and *selective* incentive will stimulate a rational individual in a latent group to act in a group-oriented way”; that is, only a clear incentive obtained through participation, and not otherwise, will provide the motivation to contribute to the group effort. This means that individuals will only act collectively to produce goods that are exclusive and solely available to those that have contributed. In other words, people will not produce or make something collectively that somebody else can then obtain without having contributed, i.e., through free-riding on the efforts of others. On such a basis, it is not surprising that CBPP in many respects has been a challenge to current thinking.

Note though that Olson explicitly excluded small groups based on family relationships and emotional ties; nor did he consider non-profit organisations and non-rival collective scientific efforts.

7. http://en.wikipedia.org/wiki/Homo_economicus; accessed August 10, 2011.

3.3 Production and management of goods through commons-regimes

A commons does not imply any specific community practises. Instead, the emphasis is on sharing, self-selection, and voluntary contributions. Excludability when applied to a commons regime will therefore relate to relationships between the community and its surrounding parties.

Traditionally, the commons concept has been related to shared natural resources and an important issue has been the management or lack of management as utilisation increases and the resource is no longer abundant but becomes rival, i.e., as the resource changes from what is effectively a public good and instead becomes a common-pool resource, cf. Table 3.1.

In such cases, conflicts have been a standard result. One of countless historical examples was the so-called enclosure in Britain whereby common land was physically enclosed and privatised over several centuries. This often meant that people were driven away so that private owners could use the land for other purposes such as large scale sheep grazing for the production of meat and wool⁸.

The rival and non-excludable common-pool resources were the topic of a very influential paper published by Garret Hardin (1968), *The Tragedy of the Commons*, which in many ways discredited the concept of the commons. Hardin describes how individuals sharing a commons-resource, e.g., a grazing field, with each acting out of self-interest trying to maximise their own benefit. As a consequence, the resource is depleted and goes to waste, in turn destroying the livelihood of the community. In Hardin's own vivid words:

Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons. Freedom in a commons brings ruin to all.

The morale is in line with Mancur Olson's view: A community will be unable to sustain itself through collective action due to free-riding, overuse, lack of organisation, and motivation. This may indeed be a realistic description concerning rival and truly non-excludable resources. In his book, *Collapse*, Jared Diamond (2004) describes the decline of several historical civilisations where overuse of natural resources was among the leading causes of their demise. Influenced by Hardin's analysis, a conventional view has become that natural resources are best administered either through privatisation or public management.

However, it turns out that many traditional societies have managed to sustain rival resources as long-term commons, thus avoiding the "tragedy". This has been extensively documented by Elinor Ostrom and co-workers at Indiana University. Over a life-time of scientific work, Ostrom (1990, 2005) has studied such commons. Based on thousands of field studies, she and her co-workers have shown that, under certain conditions, peers

8. James Boyle (2003) talks about the "second enclosure" in connection with privatisation and commodification of information and knowledge.

can indeed manage their local resources in a sustainable way that even outperforms privatised and public regimes. For this to work, she has identified eight design principles. One of these conditions is that some sort of excludability must be introduced towards external un-entitled groups. Thus the resource must become exclusive to the group, or subject to what is called a *common property regime*.

Other rules by Ostrom include securing some form of democratic governance, establishing regimes for appropriation (harvesting) and provisioning (development and maintenance), monitoring, sanctions and conflict resolution. This helps building and sustaining sufficient trust and reciprocity among the group members, which are essential to avoid the tragedy of the commons⁹. Ostrom's last design principle concerns scaling to larger networks where she advocates a multi-layered, nested approach creating a polycentric system. The idea is to form larger structures by building on working local communities. The key point is that in the absence of traditional management structures, a community must develop self governance through mutual trust and reciprocity in order to function and avoid struggle and collapse. This is not surprising in theory, but it is more surprising that it is actually achievable and a sustainable third mode of resource management, albeit not a simple one. As noted, Ostrom committed her life's work to this research, for which she received the Nobel Prize in Economics in 2009.

Turning to the non-rival goods to the left in Table 3.1, the situation changes since there is no "overuse" and no conflicts related to inherent scarcity. Our interest here is in intellectual goods. One may ask why community members would carry out the work, i.e., do the provisioning. The case of excludable goods is fairly simple also from Olson's viewpoint. Here, what is provided is exclusively for the community members and not for others, thus giving the members direct benefits. Disregarding secret societies, like the free masons, we may consider medieval guilds as an example. There, secrets of the trade, such as glass manufacturing, smithing or stone masonry, were watched over by the guild members. The knowledge was only taught to selected apprentices who would advance through the community as they gained trust through demonstration of competence and skills. Today, we still see the same practices within some communities, such as magicians or culinary chefs (Fauchart and von Hippel, 2008; Loshin, 2008). Strong norms preserve the secrets of the trades within the community; unauthorised revelations may result in strong condemnation and social exclusion.

The remaining non-excludable and non-rival case poses more of a problem from a traditional point of view. Why would community members contribute and how would they organise themselves? Members of a CBPP community participate voluntarily, and because they want to participate according to their interests and abilities. They may also want to show others what they have achieved. In short, they are part of a *gift economy*. Thus, they form a select and often resourceful group, and others can use what they produce without interfering with the community, at least as long as there are no vandals. Re-

9. In Norwegian the word *dugnad* means to participate in voluntary work in a group. A common example is the traditional *dugnad* by members of a *borettslag* (housing cooperative). However, fewer people seem to participate these days, indicating less community commitment.

member the so-called *participation inequality*: in typical CBPP communities most people only read and use, while only a few percent contribute. However, although CBPP communities are sharing freely, they are usually concerned with how their products are then used further downstream. This is specified through licences.

As indicated, CBPP communities can be studied from many angles. As a starting point, we will choose one particular view and consider scientific communities which are interesting as a kind of fore-runners in important ways. This was noted already by Raymond (1999) and is worth consideration, also because the Internet grew out of a scientific engineering community.

3.4 Scientific communities

Today, scientists are professionals employed by universities and research institutions. However, scientists also identify themselves to a large extent with the community within their field. Scientific communities are meritocracies and characterised by clear norms of attribution, openness, and trust, as well as competition and conflict.

Modern science traces its beginnings to what has been denoted as the “scientific revolution” occurring as part of the European Enlightenment. The rise of modern science was, however, not a sudden event, but emerged over a considerable time period. Some authors consider the period to be the 144 year interval from 1543, when Copernicus published his book on the heliocentric system, to 1687, when Newton published *Principia Mathematica* (Watson, 2006). During this period, a mode of rational inquiry was established based on a systematic experimental approach coupled with the crucial practice of open peer review.

The Royal Society, established in London in 1660, was instrumental in institutionalising this approach (Bryson, 2010). Manuscripts were circulated internationally for comments, read to the Society in public, and printed in their journal “*The Philosophical Transactions of the Royal Society*”. The members of the Society were a curious group interested in practical matters and carried out various experiments and demonstrations in their meetings, including animal dissections.

Most of them were men of independent means, and not affiliated with universities, which still relied on traditional scholarship. Newton’s work subsequently gave an enormous status and impetus to the natural sciences by showing how a small set of basic principles could explain a vast set of observations both terrestrial and astronomical. Newton himself somewhat modestly said:

If I have seen further, it is only by standing on the shoulders of giants.

Open peer review was invented by the Society to establish priority and credit, but was also, though unintended, a key factor leading to the explosion in creativity and innovation in the natural sciences. Science prospered simply because this is a productive and self-reinforcing way of creating new knowledge.

Scientific communities in general have certain characteristic norms which are worth noting in relation to CBPP:

Self-selection: Scientists generally select for themselves what they will work on. Most workers contribute only small bits as part of their formal degrees, and go on doing something else. However, some few choose science as a career and become professionals. Academic freedom through permanent, tenured positions is generally recognised as the best way to promote creativity and scientific progress.

Attribution and priority: Credit must always be given. In writing, this is done through citations. Claiming priority of a result that has already been published by others is a mistake that must be rectified in public. If it is done on purpose, i.e., plagiarism, it is unforgivable and would easily lead to expulsion from the community.

Incentives: Productive scientists obtain a genuine satisfaction from their work and especially if the work is carried out together with close and equally skilled colleagues who share the same strong interests.

In addition, scientific communities are meritocracies where reputation and recognition play an important role for most scientists. Research grants, for instance, are highly dependent on recognition. It can only be earned through contributions deemed significant enough by peers to warrant attribution. In this way, a particular hierarchy is established with internal quality norms. The desire for recognition and reputation can foster strong conflicts and competition. For this Watson (1968) gives a good description in his book *The double helix*, which describes the race to determine the chemical structure of DNA.

Peer review and publication: Communication between scientists continuously takes place, ranging from discussions between close and trusted colleagues to formally written publications subject to anonymous peer review. To ensure priority, all articles in reputed journals carry the date they were received by the journal as well as when they were received again after possible revisions. These publications constitute the formal communication to the wider community and to posterity, and which, if judged interesting enough by peers, will be cited. Journals come in a wide variety, some highly specialised while others are of a general nature. They have different reputation and status indicated today by the so-called *impact factor*, which is a measure of the average number of citations articles in a journal receive.

Science and IPR: Scientists have traditionally been satisfied with producing knowledge as a public good, open for everyone to apply and use. IPR and market considerations have not played a major role in scientific communities as incentives. However, with the increased commercial value of knowledge and intellectual goods, this attitude has been changing over the last decades. In 1980, the US passed the Bayh-Dole act giving universities, small businesses and non-profits the commercial rights to scientific results from publicly funded research projects. The universities then established Technological Transfer Offices (TTOs) to facilitate commercialisation. A similar law was passed in Norway in 2003, with the subsequent establishment of university TTOs.

Another issue regarding science and IPR is the copyright on scientific articles. Sci-

entists typically submit their papers for free to the best reputed journals they think may publish the work. The journals are traditionally published by commercial publishers which then, if the paper is accepted, want the copyright transferred to them. The scientists' home institutions, in their turn, subscribe to the journals often paying a steep price. This practise, considered by many as rather arcane and excludable towards other scientists especially in developing countries, is being disrupted by several so-called Open Access (OA) initiatives. For example, institutions may secure the right to self-publish their papers on the Internet. One also observes many new OA journals which distribute papers for free, but charge the authors a fee instead.

3.4.1 The Internet

The Internet grew out of an effort in the 1960s to connect mainframe university computers in the US in order to utilise available computing resources more efficiently. The work was originally financed by the US Department of Defence ARPA (Advanced Research Projects Agency) and has gone through a fascinating development. Abbate (1999) gives a highly readable account of the first decades with a focus less on technology and more on the personalities involved and the social environment. A key factor was that the developers were drawn from a scientific engineering community and given freedom to design an open, distributed and generally accessible network. Critical decisions were to use so-called packet switching implemented through TCP/IP protocol¹⁰ and base the design on the end-to-end principle, meaning that processing of content is not carried out in the network but exclusively at the sender and receiver. The result was a network that transmitted standardised data packets from sender to receiver through the links that were available, without any consideration of packet contents. In some ways, this resembles the ordinary mail system which is an infrastructure for transmitting letters or packets irrespectively of content.

The Internet is a good example of a very successful effort carried out in an open scientific mode of collaboration which in many ways is similar to CBPP. Note that at some point in the early development, it was not clear what the major applications to be built on the Internet would be. Innovations, such as e-mail, the World Wide Web and file sharing came from outside the core community.

The early spirit of the Internet has been characterised by computer scientist David Clark (1992) as

We reject: kings, presidents and voting. We believe in: rough consensus and running code.

In such a spirit, it is not surprising that security and fraud were not high on the agenda in the original Internet engineering community. Therefore, the first computer worm, released by Robert Morris in 1988 (*The Great Worm*), caused havoc, and a rethink of security and reliability.

10. http://en.wikipedia.org/wiki/TCP/IP_model; accessed September 1, 2011.

The scientists involved circulated ideas and opinions through informal notes or so-called RFCs (Requests For Comments). Today, Internet standards are developed and maintained by the Internet Engineering Task Force¹¹ (IETF) which is still an open community working through RFCs and consensus. (Some of their voting techniques may even seem a bit peculiar. Dusseault (2007) describes how *humming* is used in a meeting. The strength of a group humming sound is an indicator for which proposal to follow.)

3.4.2 “Open Science”

From what is stated above, the term “open science”¹² may seem contradictory. But science is very competitive and scientists make their careers based on what they publish. This creates a drive to secure priority and proper attribution which the traditional journals take care of, but which does not fully use the potential collaboration on the Internet.

There are several more recent developments that may lead to new practices. One is post-publication peer review, where a paper receives comments and suggestions from peers *after* publication on the Internet. The peer review is performed in public, which could be valuable in itself, and the author may respond and produce a new version. Other developments include open sharing of data such as the *Open Notebook Science*¹³. Here, even laboratory data are shared freely; and results are developed as a community effort. Note that this will also have the effect of guarding against scientific fraud. But in order to prosper, such new practices will require changes in how attribution and merit are evaluated in scientific communities. The distinguished physicist Michael Nielsen¹⁴ is a strong proponent of new practises in this area.

3.5 Commons-Based Peer Production

As in science, CBPP communities also produce non-rival and non-excludable intellectual goods and these two types of communities, as we will observe, share many other characteristics as well. Regarding the differences, science, at least pure research, often provides new knowledge without obvious applications. In CBPP, on the other hand, the goal is often more practical producing something of more immediate value to the community and society. Such goods may therefore be in more direct competition with what the market already provides. Again, FOSS is the obvious example providing software free of charge in direct competition with the software industry. Another obvious difference is that scientists are now professionals earning both a living and creating a career from what they do. This leads to pressure and competition among scientists as exemplified by the expression “publish or perish”. The volunteer participants in CBPP, on the other hand, work more on a hobby basis and can afford a more relaxed attitude.

Here, we start by repeating a fuller version of what characterises CBPP projects and communities; we also refer to Benkler (2002), and to Chapter 2.

11. <http://en.wikipedia.org/wiki/IETF>; accessed September 1, 2011.

12. http://en.wikipedia.org/wiki/Open_science; accessed September 1, 2011.

13. http://en.wikipedia.org/wiki/Open_Notebook_Science; accessed September 1, 2011.

14. http://en.wikipedia.org/wiki/Michael_Nielsen; accessed September 1, 2011.

Projects. The project must be organised so it can be carried out in a *modular* way allowing different parts to be worked on in parallel. Benkler also stresses that the set of possible tasks should be *heterogeneous*, i.e., require a range of different skills. In addition, tasks should range from the very simple to the more complex and time consuming, i.e., have different *granularity*.

Peers. A CBPP community is decentralised and the peers volunteer on a not-for-profit basis by self-selecting what they want to do. If a project can be set up as described above (i.e., being modular, with tasks of sufficient heterogeneity and granularity), it will provide incentives to a diverse community of people with different backgrounds and interests as well as differences in motivation and willingness to invest time and effort. This is critical for success since the project depends on voluntary contributions by the peers typically in their spare time, i.e., it depends on their *excess capacity*. This capacity, of course, varies widely between individuals and over time for the same person.

Integration. Integration requires both quality control to fend off incompetent or malicious contributions and ways of combining contributions into a whole. Benkler mentions four combined mechanisms for integration: 1) iterative work by a diverse community providing creativity and redundancy, 2) technical solutions (e.g., version control systems, mailing lists, wikis), 3) legal licences, and 4) norm-based social organisation with a limited re-introduction of hierarchy or market. The integration function must be low-cost or itself sufficiently modular to be peer-produced, e.g., through voting procedures.

It is interesting to speculate which types of non-rival and non-excludable goods can and cannot be produced through CBPP. Before FOSS emerged, few would have thought that quality software could be produced in this way. The community created both the norms required, the technology that was needed, and the necessary legal framework. One should therefore be careful in excluding possibilities in advance even though it could be tempting to exclude goods such as novels, symphonies, and films since these are normally produced single-handedly or in a strongly hierarchical way. However, what is called participatory media can be found on the Internet¹⁵.

Regarding CBPP communities, it is interesting to stress two interconnected observations mentioned earlier – 1) the so-called *participation inequality* and 2) the *decisive role of merit through participation*.

A common rule of thumb is the 20/80 rule (the Pareto principle), which states that 20 percent of those involved in an activity will do about 80 percent of the work. On the Internet, one typically observes a more extreme version of this principle denoted the *participation inequality* or the 90-9-1 rule¹⁶: only 1% of the community – the super-contributors – create content, 9% edit and modify it, and 90% – the “lurkers” in the Internet jargon – just view and read. Actual communities, of course, show variations from these numbers but

15. http://en.wikipedia.org/wiki/Participatory_media; accessed September 5, 2010.

16. <http://en.wikipedia.org/wiki/90-9-1>; accessed August 4, 2010.

they do indicate what to expect. With the non-rival nature of CBPP, one would certainly expect a relatively large number of “lurkers” but such highly skewed ratios may seem surprising, not least given the attention surrounding Web 2.0. But this observation does stress that there is always a potential for increasing the active part of the community by cultivating the inactive groups. The virtuous, anti-rival circle is one where more people visiting the project increases its attractiveness leading to further participation.

The super-contributors will tend to become the senior members of the community with the most trust and authority. These, in time, will often form the upper echelon of an informal or formal hierarchy based on merit. Thus, as in science, a meritocracy develops where a person can only gain status and seniority within the community through contributions that are deemed important and valuable enough by the peers.

Within FOSS, the role of meritocracy was explicitly formulated already by Steven Levy (1984) in his book *Hackers: Heroes of the Computer Revolution* as one of seven principles describing the hacker ethics. His principle number 5 states: *Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race or position*. The other principles in his book stress a practical hands-on approach to computing, freedom of information, a mistrust of authorities, and computer programming as an art creating beauty.

A question often raised is why people participate in CBPP communities where most of the others are strangers. Since this is often a hobby based on “excess capacity” or “cognitive surplus”, there is seldom monetary incentives, although participation and the resulting visibility could lead to future work opportunities. Idealism and ideology certainly play a role as exemplified by the Free Software Foundation. More mundane reasons could be the need for developing or improving a certain tool to ease everyday tasks. It is evident that incentives similar to those that motivate scientists are important such as the simple joy and genuine satisfaction from the work, cf. the book about Linus Torvalds: *Just for fun: The story of an accidental revolutionary* (Torvalds and Diamond, 2001).

This is strengthened by working closely with other people sharing similar interests, even though they initially are strangers. As the contribution grows, reputation and recognition in the community also grows. In an on-line investigation by Andy Oram on the motivation for writing free software documentation, community building and the personal benefits of learning through teaching came out on top (Oram, 2007).

Knowledge about why people contribute is obviously important in designing a community. In this respect we can also draw on Ostrom (2005)’s design principles. As noted, these were developed for rival natural resources managed collectively (Common Property Regimes – CPRs). But they do provide important empirical guidelines as to how human communities work best to provide a commons based on sustainable trust and reciprocity among group members. A complete list is:

1. Clearly defined boundaries (effective exclusion of external un-entitled parties). This is not applicable to CBPP.
2. Rules regarding the appropriation (harvesting) and provision (development) of com-

mon resources are adapted to local conditions.

3. Collective choice arrangements allow most resource appropriators to participate in decision-making processes.
4. Effective monitoring by monitors who are part of, or accountable to, the appropriators.
5. There is a scale of graduated sanctions for resource appropriators who violate community rules.
6. Mechanisms for conflict resolution are cheap and easy to access.
7. The self-determination of the community is recognised by higher-level authorities.
8. In the case of larger common-pool resources (CPRs) organisation in the form of multiple layers of nested enterprises with small local CPRs at the base level.

The question of exclusion towards external parties is not relevant for non-rival CBPP projects. Neither will appropriation have to be regulated nor provisioning since that takes place on a voluntary basis. However, the other rules are relevant. Most functioning CBPP communities will already have similar rules in place. But the guidelines should be valuable when serious problems occur or one wants to establish new communities.

As emphasised by Benkler (2002) participation in CBPP communities strengthen civil society and has intrinsic democratic value. But it is well worth reflecting on why CBPP may compete well with commercial actors. In the production of intellectual goods, the critical factors are human motivation, knowledge, and creativity. This is not necessarily simple to harness in a hierarchy with formal lines of authority and bureaucratic constraints. It requires organisational costs and the result is often a lack of efficiency. The same is true for a company in the market where costs and risks are associated with decisions such as hiring or buying and outsourcing when human factors are the critical issue, cf. Benkler (2002)¹⁷. The result is a certain rigidity of structures, which has consequences that are well captured (albeit humorously) by *Conway's law* (Conway, 1968):

Organisations which designs systems are constrained to produce designs which are copies of the communication structures of these organisations.

CBPP offers advantages in these respects since people are driven by internal motivation and self-select what they want to do. The potentially strong benefits are well described by a quote from the British philosopher Bertrand Russell (1933): *Skilled work, of no matter what kind, is only done well by those who take a certain pleasure in it, quite apart from its utility, either to themselves in earning a living or to the world though its outcome.* However, there are also negative sides related to self-selection, including incompetence and vandalism. To correct for this, CBPP crucially makes use of peer review.

17. Similar points were emphasised early by management professor Peter Drucker, who coined the term "knowledge worker" already in 1959; see http://en.wikipedia.org/wiki/Peter_Drucker; accessed August 4, 2010.

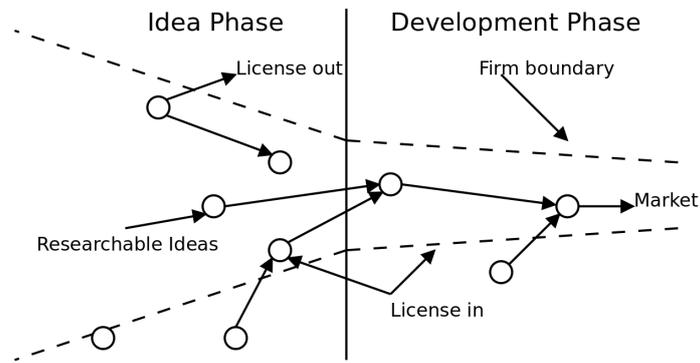


Figure 3.1. The Open Innovation Model.

As noted, legal copyright licences play a key role in CBPP. This was discussed in Chapter 2 and a full treatment is given in Chapter 7. Suffice it to say here that CBPP communities are strongly concerned with and careful about copyright and the licence to be used. The GPL created by Richard Stallman set the precedence for the Creative Commons licences which generally give communities both the freedom and protection they want.

A possible conclusion is that both scientific and CBPP communities have arrived at the most productive modes of creative and knowledge-oriented collaboration. Science came first, and one may argue that scientific communities have strongly influenced CBPP. Such modes of collaboration have now proven extremely productive in whole new domains. This may be the way large communities of humans sharing the same interests and having various degrees of skills and commitment, can work most productively together.

3.6 Open Innovation

Earlier in this compendium we mentioned the innovative and disruptive aspects of FOSS and CBPP in relation to the two traditional modes of production: the market and the public sector. *Crowdsourcing*, referring to the practice of letting a project or task be open to contributions from anyone, is a critical aspect here. This is not a new idea in innovation theory but has gained momentum in recent years following the success of FOSS and CBPP. The study of innovation is a large field in itself, and one which we will not delve deeply into here, but instead refer to a standard text such as Rogers (2003).

The term *open innovation* – see Figure 3.1 – was coined by Chesbrough et al. (2006) to describe how companies can, on the one hand, open up and more systematically draw on ideas from the outside and, on the other hand, initiate start-ups or new collaboration based on their own ideas that are outside their main domain.

Note that open innovation, as described by Chesbrough et al. (2006), only applies to how companies can better harvest outside ideas or spin off new companies themselves. It does not imply an open sharing of ideas similar to FOSS. On the contrary, the setting is market-based, and therefore securing and respecting IPR, in particular patents, are central to the concept. Open innovation is related to the idea of *user innovation* or *user-driven innovation* promoted by von Hippel (2005). Here the emphasis is on the users of

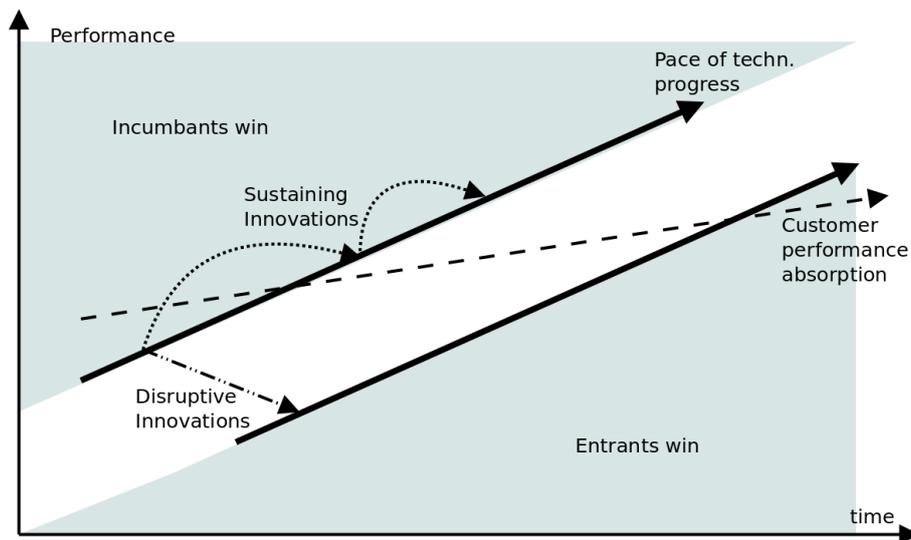


Figure 3.2. Disruptive innovation model, after Christensen (2003).

a product as a source of ideas for improvements.

Given FOSS and CBPP one would not be too surprised that open innovation and user innovation can be effective. However, innovation theory was traditionally based on a one-way, three-stage linear model¹⁸. Here, the starting point is inventions (basically new ideas) that later are turned into practical innovations diffusing into the society. Although such simple thinking is not mainstream any more, open innovation and user innovation would still, until recently, seem radical to many.

A good example of open innovation is InnoCentive¹⁹. This is a private company facilitating crowdsourcing competitions.²⁰ Organisations having problems they want solved – i.e., the *Seekers* – can post these on the Internet through InnoCentive, which has registered an open community of *Solvers*. Companies will normally offer a cash prize for what they consider the best solution. InnoCentive works in diverse areas including chemistry, life science, physics, and business and entrepreneurship. Issues related to confidentiality and IPR are handled by InnoCentive, which has achieved considerable success and can count *The Economist* and the scientific journal *Nature* among its partners.

It is worth noting that crowdsourcing competitions are not a new phenomenon. A well known historic example is the *Longitude prize*²¹ established by the British parliament in 1714. The problem at the time was to establish a ship's longitude at sea. The solution involved the construction of a sufficiently accurate chronometer and over the years over £100,000 were paid to various "solvers". What InnoCentive has done is to systematise such an approach using web technology and proven to the business community, that this is a viable way to innovate.

18. http://en.wikipedia.org/wiki/Linear_model_of_innovation; accessed March 4, 2012

19. <http://www.innocentive.com>; accessed March 4, 2012

20. Other examples for companies facilitating crowdsourcing open innovation include clickworker.com; accessed March 7, 2012 and atizo.com; accessed March 7, 2012.

21. <http://en.wikipedia.org/wiki/Longitude>; accessed March 4, 2012.

Christensen (2003) argues strongly in favour of having a business model that constantly emphasises the role of new ideas from outside the company. This is to guard the company against what he calls external *disruptive innovations*. These are innovations that rapidly reduce the cost of goods or services in an industry, and at the same time, delivers an acceptable performance. He describes two fundamental types of disruption, namely the low-end disruption, and the new markets disruption. This model is shown in Figure 3.2.

References

- Janet Abbate. *Inventing the Internet*. MIT Press, 1999.
- Yochai Benkler. Coase's Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal*, 112:369–446, 2002.
- Yochai Benkler. *The Wealth of Networks*. Yale University Press, October 2007. ISBN 0300125771, 9780300125771.
- James Boyle. The Second Enclosure Movement and the Construction of the Public Domain. *Law and Contemporary Problems*, 66:33–74, 2003. doi: 10.2139/ssrn.470983. URL <http://ssrn.com/abstract=470983>.
- James Boyle. *The Public Domain: Enclosing the Commons of the Mind*. Caravan, 2008. ISBN 9780300137408.
- Bill Bryson, editor. *Seeing further. The story of science and the Royal Society*. Harper, 2010. ISBN 9780007302567.
- Henry William Chesbrough, Wim Vanhaverbeke, and Joel West. *Open Innovation: The New Imperative for Creating And Profiting from Technology*. Oxford University Press, August 2006. ISBN 0199290725, 9780199290727.
- Clayton M. Christensen. *The Innovator's Dilemma: The Revolutionary Book that Will Change the Way You Do Business*. Harper Paperbacks, January 2003. ISBN 0060521996.
- David D. Clark. A cloudy crystal ball – visions of the future. page 551. IETF, July 16 1992. URL <http://ietf.org/proceedings/prior29/IETF24.pdf>. Presentation given at the 24th Internet Engineering Task Force.
- M.E. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.
- Jared Diamond. *Collapse: How Societies Choose to Fail or Succeed*. Viking Adult, 1 edition, December 2004. ISBN 0670033375.
- Lisa Dusseault. Notes on IETF rough consensus. IETF Internet Draft, December 2007. URL <http://tools.ietf.org/html/draft-dusseault-consensus-00>.
- Emmanuelle Fauchart and Eric von Hippel. Norms-based intellectual property systems: The case of french chefs. *Organization Science*, 19(2):187–201, 2008. ISSN 1526-5455. doi: <http://dx.doi.org/10.1287/orsc.1070.0314>.

- G. Hardin. The tragedy of the commons. *Science*, 162(3859):1243–48, 1968.
- Michael Heller. *The Gridlock Economy: How Too Much Ownership Wrecks Markets, Stops Innovation, and Costs Lives*. Basic Books, 1ST edition, July 2008. ISBN 0465029167.
- Charlotte Hess and Elinor Ostrom. *Introduction: An Overview of the Knowledge Commons, In Understanding Knowledge as a Commons*. MIT Press, 2007. ISBN 0-262-08357-4.
- Josh Lerner and Mark Schankerman. *The Comingled Code, Open Source and Economic Development*. MIT press, 2010.
- Lawrence Lessig. *Free Culture: The Nature and Future of Creativity*. Penguin (Non-Classics), 2005. ISBN 0143034650.
- Steven Levy. *Hackers: Heroes of the Computer Revolution*. Penguin (Non-Classics), 1984.
- Jacob Loshin. Secrets Revealed: How Magicians Protect Intellectual Property without Law. *Law and Magic: A Collection of Essays*, July 2008. URL <http://ssrn.com/abstract=1005564>.
- Mancur Olson. *The logic of collective action*. Harvard University Press, 1965. ISBN 0674537513, 9780674537514.
- Andy Oram. Why do people write free documentation? results of a survey. web, June 14 2007. URL <http://onlamp.com/pub/a/onlamp/2007/06/14/why-do-people-write-free-documentation-results-of-a-survey.html>.
- Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, 1990. ISBN 0521405998.
- Elinor Ostrom. *Understanding Institutional Diversity*. Princeton University Press, 2005. ISBN 0691122385.
- Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, Sebastopol, California, 1999.
- Everett M. Rogers. *Diffusion of Innovations*. Free Press, 5th edition, August 2003. ISBN 0743222091.
- Bertrand Russell. *The Conquest of Happiness*. Liveright, 1933. ISBN 0871401622.
- Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, Massachusetts, 2002.
- Linus Torvalds and David Diamond. *Just for fun: The story of an accidental revolutionary*. HarperCollins, 2001. ISBN 0066620724.
- Eric von Hippel. *Democratizing Innovation*. MIT Press, 2005. ISBN 0262002744, 9780262002745.

James D. Watson. *The Double Helix: A Personal Account of the Discovery of the Structure of DNA*. Touchstone, 1968. ISBN 074321630X.

Peter Watson. *Ideas: A History of Thought and Invention, from Fire to Freud*. Harper Perennial, October 2006. ISBN 0060935642.

Steven Weber. *The Success of Open Source*. Harvard University Press, April 2004. ISBN 0674012925.

4 Free and Open Source Software



by Wolfgang Leister

According to Wikipedia (2010)¹, *free and open source software, also F/OSS, FOSS, or FLOSS (free/libre/open source software) is software that is liberally licensed to grant the right of users to study, change, and improve its design through the availability of its source code.*

Free and open source software (FOSS) covers both *free software* and *open source software*. While both follow similar development models they have their roots in different cultures, philosophies, and business models. In this current chapter we look at the different perspectives of FOSS, from a theoretical, practical, and pragmatic point of view; we cover free software with its focus on freedoms and open source as a development model-centric approach. We also cover history, social context and culture, licenses, business models, development processes, software and distributions. We motivate this in Figure 4.1. Quality metrics for FOSS will be discussed in Chapter 5 separately.

Many successful FOSS projects have an impact above the software as such, as they influence an ecology of other goods. As an example, the typesetting system \TeX and the font design system METAFONT developed by Donald F. Knuth (1984, 1986), fostered the design of freely available fonts. In the context of CBPP projects many tools have been developed as FOSS, such as editing and rendering software for OpenStreetMap².

4.1 FOSS Concepts

The concept of FOSS is varying, depending on philosophical, commercial, or cultural viewpoints. However, there seems to be a common understanding of its definition, resulting in the one given in Wikipedia (2010). Each piece of software that is created by an author, and thus is subject to copyright, needs to be licensed in order to be used. The license is the vehicle for the software to become useful to others. In this license the different philosophical viewpoints may be encoded.

The most general definition of FOSS is given by the Open Source Initiative (OSI)³ which present a list of ten criteria (Perens, 1999), as shown in Frame 4.1. This list can be seen as a common denominator for FOSS.

1. We cite this definition from Wikipedia since this encyclopedia is based on the same principles as FOSS. The page where this definition is taken from is heavily discussed, and we therefore assume that the definition of FOSS given here is the result of a community effort.

2. See <http://www.openstreetmap.org>; accessed September 24, 2010.

3. See www.opensource.org; accessed September 24, 2010. The definition of FOSS and the rationales behind these terms can be found at www.opensource.org/docs/definition.php; accessed September 24, 2010.

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. **Free Redistribution.** The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
2. **Source Code.** The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.
3. **Derived Works.** 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.
4. **Integrity of The Author's Source Code.** The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
5. **No Discrimination Against Persons or Groups.** The license must not discriminate against any person or group of persons.
6. **No Discrimination Against Fields of Endeavor.** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
7. **Distribution of License.** The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
8. **License Must Not Be Specific to a Product.** The rights attached to the program must not depend on the program's being part of a particular software distribution. 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.
9. **License Must Not Restrict Other Software.** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.
10. **License Must Be Technology-Neutral.** No provision of the license may be predicated on any individual technology or style of interface.

Source: © [Opensource.org](https://opensource.org/), CC-BY.

Frame 4.1. The ten rules of the OSI

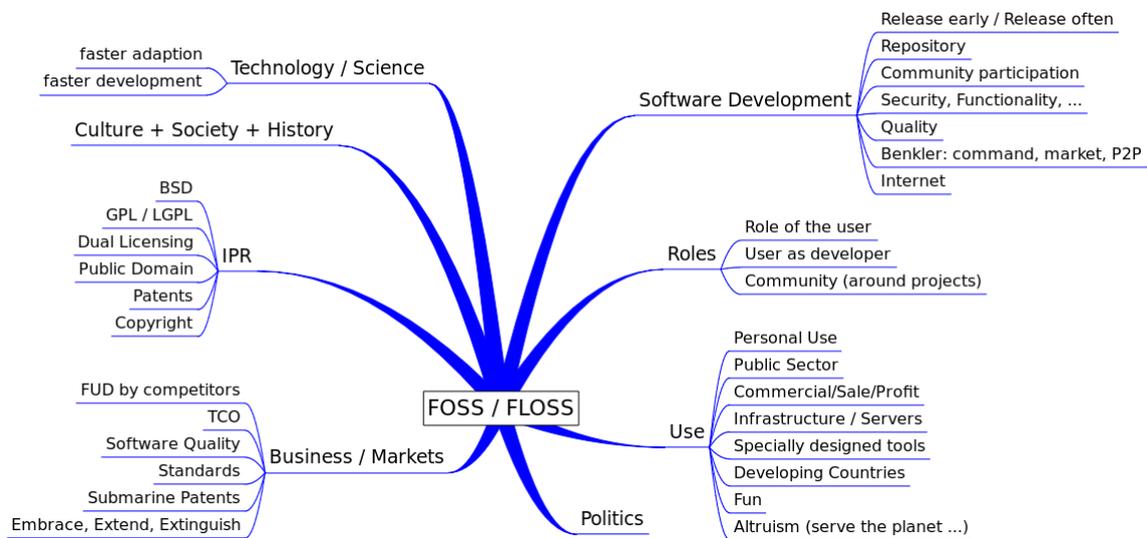


Figure 4.1. A Mindmap of FOSS and its relations

Freedom 0: The freedom to run the program, for any purpose.

Freedom 1: The freedom to study how the program works, and change it to make it do what you wish.

Freedom 2: The freedom to redistribute copies so you can help your neighbour.

Freedom 3: The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes.

adapted from <http://www.gnu.org/philosophy/free-sw.html>.

Frame 4.2. The four software freedoms

Many FOSS licenses fulfil the ten OSI criteria, in fact, the OSI has approved over eighty compliant licenses⁴. Some of them are liberal licenses, i.e. only attribution to the authors is required, while other licenses have more strict conditions, e.g., restrictions on what kind of licenses derivatives must have.

The Free Software Foundation (FSF)⁵ defines four freedoms for software, as shown in Frame 4.2. The freedoms to run, study, redistribute, and distribute modified versions of the software, also called the four essential freedoms⁶, must always be fulfilled, also for derivatives. To enforce these freedoms, the FSF use the copyleft, a concept introduced by Stallman (2009) around 1985.

Stallman and the FSF promote *freedom* as the main goal, for which access to the source code is a precondition to achieve Freedoms 1 and 3. In contrast, the OSI talks of *open source*, that is the access to the source code, only. Note that the ten criteria of the OSI are fulfilled for software licenses by the FSF following the four freedoms, but not the reverse.

4. See opensource.org/licenses/; accessed February 23, 2012.
 5. See www.fsf.org; accessed February 23, 2012.
 6. See <http://www.gnu.org/philosophy/free-sw.html>; accessed August 8, 2010.

Feller and Fitzgerald (2000) present an overview, taxonomy, and analysis of FOSS and its development process. In this taxonomy, FOSS is compared to products that are commercial, trial software, use-restricted, shareware, freeware, royalty-free binaries, and royalty-free libraries. The following software categories can usually **not** be considered as FOSS, since the source code is not available according to the aforementioned rules:

Freeware: The software can be used without costs. However, as long as the source code is not freely available, freeware is considered proprietary software. The freeware author usually restricts one or more rights to copy, distribute, and make derivative works of the software. The software license may impose restrictions on the type of use, e.g., personal, individual, non-profit, non-commercial, or academic use.

Nagware: as freeware, but dialog boxes remind the user that these boxes will disappear after a fee is paid.

Adware: as freeware, but the display of advertisements is used as means of “payment”.

Crippleware: as freeware, but some functionalities are restricted in the free version. After payment of a fee the restrictions are removed.

Shareware: Quite often, shareware is developed originally by enthusiasts who market their program using the shareware mechanism to earn some money. While the source code is closed, the program can be used for a trial period for free, but is subject to payment of a fee after that, if the user still wants to use the software. From a licensing-perspective, shareware is proprietary software.

4.2 Historical and Societal Context

Raymond (1999) presents the history of FOSS from his own perspective. He divides the time scale into several eras, beginning in the 1960's, to about the year 2000. To show the historical perspective, we compile a time line inspired by Raymond:

Until about 1960: The Real Programmers. The price of computers in those days were so high that they were not affordable for personal use, and only research labs (universities) and companies were able to purchase them. Raymond describes the rules of a traditional scientific community and those of large businesses at the time.

1960-1970: The Early Hackers. Mini machines, such as the PDP were used at universities, and operated by the research groups. In this environment the early hackers formed a community fostering an exchange of programs and software. The MIT scientists followed different ideas than the company DEC behind PDP, and developed the ITS (Incompatible Timesharing System) with its community of projects fostering LISP and eventually *emacs*.

The jargon files (Raymond, 1996; Steele and Raymond, 2000) are a witness from this cultural context. The ARPANET was developed from Department of Defense funding, and

created the basis for the Internet. The early hackers were heavily involved in this development process. The Internet Engineering Task Force (IETF), who still govern the Internet today, are inspired from these work principles.

1970-1980: The Rise of Unix. Ken Thompson at Bell labs was involved with the development of Multics which has common ancestry with the ITS. Multics was not a success, and Thompson started to elaborate his ideas with an embryonic version of Unix together with Dennis Ritchie, the inventor of the C programming language. Being able to write the OS in C, gave an important advantage, and Unix spread with the company AT&T, despite of the lack of any formal support for it.

1980-1985: The End of the Elder Days. The cultures around the ITS and LISP merged with the cultures around C and Unix, favouring Unix due to its portability. Richard M. Stallman, a.k.a. RMS, started to work on the GNU operating system⁷. At the same time microcomputers using the inexpensive Motorola 68000 processor were developed, followed by the foundation of SUN Microcomputers by Stanford and Berkeley scientists, using the BSD⁸ flavour of Unix.

1985: The Free Software Foundation. RMS founded the Free Software Foundation (FSF) with the goal to develop a free version of a UNIX operating system. RMS is the creator of the *emacs* text editor, and campaigned against the commercial lab culture. In the course of this effort many GNU system tools, such as the *gcc*, were developed. RMS also developed the GNU Public License (GPL)⁹, which has its basis in the concept of the copyleft.

Stallman (2009) promoted freedom as the main ingredient; however, Stallman is not occupied with addressing businesses issues. In fact, the GPL is agnostic to how business with GPL-based software is done, as long as the terms of the GPL are followed; this includes that the code is freely available, except reasonable costs for transport, and that the other freedoms are fulfilled. See also Williams (2009).

1985-1990: The Proprietary Unix Era. Several proprietary flavours of Unix were developed, such as the AT&T System V, IRIX, HP-UX, Solaris, and AIX, all claiming their share of the market. An interesting observation is that the X windows system, developed at the MIT, and given away for free won the market for graphical window systems over the proprietary Suntools. Note that the competitors, namely MS-DOS and Apple/Mac, as well as Atari and Amiga, only to some extent were able to create a community.

7. GNU stands for the recursive acronym GNU's Not Unix.

8. BSD stands for **B**erkeley **S**oftware **D**istribution.

9. See Section 4.3.

1990-1995: The early free Unixes. HURD, the operating system promoted by GNU never arrived. In 1991, the Unix derivative Minix was created by Andrew Tannenbaum as part of his operating system teachings. At the same time, in 1991, the Finnish student Linus Torvalds started to develop Linux, while other free Unix-like systems, such as BSD (by William and Lynne Jolitz), FreeBSD (Hubbard, 1995-2010), NetBSD, and OpenBSD emerged. The latter used the BSD License⁹, while Linux 1.0 was released in 1994 under the GPL.

Torvalds and Diamond (2001) tell the history of Linux, and the socio-cultural field around its development under the sub-title *The story of an accidental revolutionary*. Motivations for his actions are based on *survival – social order – entertainment*. Many characteristics of the Linux development process are much more pragmatic than those from the FSF philosophy. Eventually, both developments complement each other.

1995-2000: The great web explosion. Linux distributions like Slackware, SLS (Soft-landing), S.u.S.E, DLD, RedHat, Debian, Knoppix, and Ubuntu were commonly available on CD-ROM¹⁰. At the same time, the *great web explosion* made it possible to cooperate in creating software over the Internet. Linux became more “fit for business”, and was increasingly used in business-critical systems, especially on the servers. Contemporarily, business strategies for FOSS were discussed by Behlendorf (1999). The development methodology of the Linux kernel, often characterised with the slogan *Release early, release often*, was different from the GNU project (Raymond, 1999).

Eric S. Raymond, a.k.a. ESR, became a spokesman for open source as a business model (DiBona et al., 1999) as he founds the Open Source Initiative (OSI)¹¹. ESR has been involved in the development of software since the seventies, and has a more pragmatic approach than RMS. He combines the freedoms of FOSS with building businesses, branding the term *Open Source*.

The Halloween Documents. In the last week of October 1998, a confidential Microsoft memorandum on their strategy against Linux and FOSS was leaked to Eric S. Raymond who published and annotated this document, and also several related documents (Raymond, 1998). The original documents were authored by Vinod Valloppillil and Josh Cohen. Raymond gives credit to Valloppillil and Cohen for authoring remarkable and effective testimonials to the excellence of Linux and FOSS in general. Over time, the *Halloween Documents* consist of eleven documents.

The new millennium. FOSS is present in most areas of software production. Diverse Linux distributions are used both in the server market, and for personal applications. Web servers using *Apache*, browsers using *Firefox*, and office suites, e.g., *OpenOffice*, are

10. The distributions could be purchased, but were often available as attachments to computer magazines. These distributions also could be downloaded for those with enough bandwidth on the Internet.

11. see www.opensource.org; accessed September 24, 2010.

widely used. For most applications there is usually a FOSS counterpart. Even in mobile phones, digital TV receivers, and other gadgets there might be FOSS products involved – especially for the markets in the third world, where FOSS based mobile phones are developed for the masses.

The Culture of Commercial Software Development. Despite the claims some authors that commercial software developers do not have an own culture, several books have been published that witness to the opposite. There are several books and musings about the traditions in the “proprietary” camp, e.g., at Microsoft. Examples are: Edstrom and Eller (1999), Gates and Hemingway (2000), Gates et al. (1995), Coupland (1996), Wallace (1998a), Gates (2001), Wallace (1998b), and Wallace and Erickson (1993).

4.3 Software Licenses

A license is a permission under intellectual property laws to authorise the use of content from a licensor to a licensee. All software used by a third party, whether in source or in compiled form, needs to be under a license when used; this spares the licensee from an infringement claim brought by the licensor. This is a consequence of copyright laws, which were developed since the eighteenth century¹², and which principles are still in use today.

The license defines the terms for the distribution and use of the software. The copyright holder can choose between a variety of licenses, both commercial or open source. FOSS is released under licenses that are compliant with the criteria published by the Open Source Initiative (OSI)¹³, as outlined in Section 4.1. In Chapter 6, we show how content other than software can be licensed following the paradigms of the open-movement.

The copyright holder can perform the following actions in order to make use of his or her rights: *a)* waive all rights to the software, which has the same consequences as if the software was in the *public domain*; *b)* license the software under an appropriate license; *c)* transfer the copyrights to others, thus passing on the copyright. A copyright holder can license software multiple times within the restrictions of copyright laws, unless a separate contract or an exclusivity clause prohibits this; this principle is called *dual licensing*. Note that only the copyright holder can dual-license software. Some licenses are non-revocable – once granted, these cannot be terminated.

4.3.1 Public Domain

Software in the public domain is not covered by any copyright. The reasons for this may include: *a)* the copyright has expired; *b)* the copyright holder has waived his or her copyrights; *c)* the copyright holder is considered unknown and cannot be contacted;¹⁴ *d)* the

12. See also Section 3.1.1 of Chapter 3.

13. A commented overview of licenses for FOSS can be found at GNU.org; accessed September 24, 2010 (GNU, 2002), opensource.org; accessed September 24, 2010.

14. Note that it must be made evident that the copyright holder is unknown. Note also that there might be heirs of a creator who can hold the copyright.

software can be considered as general knowledge. The term *public domain* does not denote a license as such. Software in the public domain can be used by everybody for any purpose, even without the obligation to attribute the original author.¹⁵

4.3.2 BSD-style

The BSD-style licenses, first used in the Berkeley Software Distribution (BSD)¹⁶ permit that the software can be copied, modified and incorporated into both open- and closed-source software. This license requires that credits to the authors of code within the source code and documentation are intact, and that the original author cannot be sued. Additionally, the original author's name cannot be used to advertise the derivative software. The BSD license permits that the program code can be distributed in closed form, not requiring that improvements are coming back to the developer community.

While it is legally possible to create a proprietary version of BSD-licensed software, this comes at a cost, since a company doing this needs to employ staff that maintains this software. Improvements and new features created by the developer community of the FOSS project are not automatically for the benefit of the commercial branch of the software. In practice, when not enough resources are put into the proprietary branch of this software, these will have fewer features over time, thus losing market value.

4.3.3 GPL/LGPL

The GPL (GNU General Public License)¹⁷ is designed to keep software free, i.e., to give the programmer the opportunity to share and change software. In addition to access to the source code, and author attribution, the GPL requires that derivative work is made available under the same conditions as the original.

In the case of the GPL, derivative work includes combining the software with other software, including linking of programs. Therefore, there are some restrictions on how to use software licensed under the GPL with respect to combining this software with proprietary software. This principle creates what often is called the "viral" behaviour: Software under the GPL cannot be combined with non-GPL software without this software also being under the GPL.

In order to avoid problems for businesses using FOSS products each product should be checked for compliance with respect to the licenses used. The implications for breaches of compliance can be handled by the courts, and can have an impact on the license to be used for a software product as a whole.

The GNU Lesser General Public License (LGPL)¹⁸ allows that software can be combined with other software by linking, but has in all other respects the same conditions as the

15. Note, however, that it is morally a good thing to give credit to the originator also if legally not required.

16. See www.bsd.org; accessed September 24, 2010.

17. See <http://opensource.org/licenses/gpl-2.0.php>; accessed September 24, 2010 (GPLv2) and <http://opensource.org/licenses/gpl-3.0.html>; accessed September 24, 2010 (GPLv3).

18. See <http://opensource.org/licenses/lgpl-2.1.php>; accessed September 24, 2010 (LGPLv2) and <http://opensource.org/licenses/lgpl-3.0.html>; accessed September 24, 2010 (LGPLv3).

GPL. The LGPL is typically used to license libraries, so that these can be combined (i.e., dynamically linked)¹⁹ with all types of software.

The version 3 of the (L)GPL (GPLv3)²⁰ addresses problem areas that have arisen since the publication of its predecessor, the GPLv2. The GPLv3 requires that the licensed software is either patent-free, or the patents can be implemented on a royalty-free basis. The GPLv3 also includes a clause that is designed to avoid restrictions of software imposed by the Digital Millennium Copyright Act (DMCA), the European Union Copyright Directive, and similar laws. This is further outlined in Section 4.8.

There are licenses that build on the GNU GPL, but remove certain terms relating to patents or other details. Examples for these licenses include the Eclipse Public License²¹ (EPL) and the Common Public License (CPL). Note that these licenses are not compatible with the GNU GPL.

4.3.4 Dual Licensing

The copyright holder can license software with several licenses at the same time. This practice is often used for commercial branches of a software, or there might be patent problems or incompatibilities with parts of the software. In general, the use of multiple licenses must be carefully considered from case to case.

An example of multiple licenses is the Mozilla Public License (MPL)²², which was developed to cope with the business situation Netscape was in the middle of the 1990ies. Today, Mozilla and attached projects use a tri-license²³ consisting of the MPL, the GPL, and the LGPL. Licenses that build on the Mozilla Public License include the CDDL²⁴ (Common Development and Distribution License) by Sun Microsystems.

Dual licensing can only be used by the copyright holder. Therefore, companies using the dual licensing model will require contributors to transfer their copyrights to them. In these cases, both a FOSS version and a commercial version with additional capabilities can co-exist, using, e.g., the *freemium* business model²⁵.

4.3.5 License Assignment

With a license assignment, the author transfers his or her rights to another party, thus giving up the rights. This can be done for various reasons for both author and receiver of the license. Companies can require license assignment from contributors to make the *open core* business model possible. Open core is a special case of the freemium business model, as explained in Section 4.7.

19. The LGPL has a stipulation that people can re-link with own libraries, and therefore must be dynamically linked.

20. See <http://opensource.org/licenses/gpl-3.0.html>; accessed September 24, 2010.

21. See http://en.wikipedia.org/wiki/Eclipse_Public_License; accessed August 25, 2011.

22. See <http://www.mozilla.org/MPL/>; accessed December 22, 2011.

23. Note that abandoning the MPL would require an effort for re-licensing.

24. See <http://en.wikipedia.org/wiki/CDDL>; accessed August 25, 2011.

25. See Section 4.7.

Some open source projects require assignment of copyright for the purpose of defending that copyright, e.g., in the case of court trials when prosecuting breaches of the GPL. In order to prevent possible use of the open-core practices in the case of changes of ownership, some organisations include clauses in their assignment to prohibit open core practices, such as the Fiduciary License Agreement by the Free Software Foundation Europe.²⁶

4.3.6 Re-Licensing

Incompatibility in the licenses, and multiple licenses often require that a software needs to be re-licensed, which is tedious work. To achieve re-licensing, the entire software needs to be checked for contributions, and all contributors need to give their consent for the new license, or the contribution needs to be licensed so that it can be used even in the eventuality of a license change. Some organisations ask contributors to re-assign copyright to them in order to ease the process of possible re-licensing.

4.3.7 Shared Source

Shared source²⁷ is an initiative that covers some of Microsoft's legal mechanisms for software source code distribution. Microsoft's Shared Source Initiative, launched in May 2001, includes licenses to view and/or use the source code subject to certain eligibility criteria. The spectrum of licenses contains open source and free licenses, closed source licenses, shared source programs, and commercial licenses. The shared source licenses include:

Ms-PL: The *Microsoft Public License* allows for distribution of compiled code for either commercial or non-commercial purposes under any license that complies with the Ms-PL. Redistribution of the source code is permitted only under the Ms-PL. The Ms-PL is approved by OSI, and a free license according to the FSF, but not compatible with the GNU GPL.

Ms-RL: The *Microsoft Reciprocal License* allows for distribution of derived code so long as the modified source files are included and retain the Ms-RL. Single files that are not part of the Ms-RL licensed material can be licensed differently. The Ms-RL is approved by OSI, and a free license according to the FSF, but not compatible with the GNU GPL.

Ms-RSL: The *Microsoft Reference Source License* makes the code available to view for reference purposes only. Developers may not distribute or modify the code. The Ms-RSL is non-free, and not OSI-approved.

Ms-LPL: The *Microsoft Limited Public License* grants the rights of the Ms-PL only to developers of Microsoft Windows-based software. The Ms-LPL is non-free, and not OSI-approved, since it is not technology-neutral.

Ms-LRL: The *Microsoft Limited Reciprocal License* grants the rights of the Ms-RL only to developers for a Microsoft Windows platform. The Ms-LRL is non-free, and not

26. See <http://fsfe.org/projects/ftf/FLA.en.pdf>; accessed February 17, 2012.

27. See http://en.wikipedia.org/wiki/Shared_source; accessed August 25, 2011.

OSI-approved, since it is not technology-neutral.

Ms-ESLP: The *Microsoft Enterprise Source Licensing Program* gives enterprise customers viewing access to some parts of some versions of the Microsoft Windows operating systems, without allowing modifications. The Ms-ESLP is non-free, and not OSI-approved.

Ms-WAP: The *Microsoft Windows Academic Program* provides universities with concepts and selected source code for teaching and research. The Ms-WAP is non-free, and not OSI-approved.

Ms-GSP: The *Microsoft Government Security Program* gives participating governments access to the source code for current versions of selected software. The Ms-GSP is non-free, and not OSI-approved.

MS-MVPSLP: The *Most Valuable Professionals Source Licensing Program* Microsoft makes source code available to members the Microsoft developer community, for debugging and support purposes, but not as an aid to develop a commercial product. The Ms-GSP is non-free, and not OSI-approved.

Ms-SSCLI: The *Microsoft Shared Source Common Language Infrastructure* licensing permits non-commercial modification and distribution of the source code, as long as the original license is included.

4.3.8 FOSS Compliance

FOSS compliance means that developers using FOSS must observe all copyright notices, and satisfy all license obligations for the software they are using. While the use of proprietary software is often negotiable, the FOSS licenses are not. This means that a company cannot solve licensing issues by buying a commercial license, unless dual licensing is possible. When using FOSS, companies need to deal with many different licenses that must be in accordance to the company's policies and goals in addition. If companies do not comply, license agreements, such as the GNU GPL, may become void, and law suits, delays in introducing a product, or a bad reputation might be consequences.

A set of organisational rules and structure are suggested by Haddad (2009). As building blocks for compliance management in enterprises he uses a team called open source review board (OSRB) including legal, technical and administrative personnel, policies, processes, tools, portals, training and guidelines, as well as 3rd party software due diligence²⁸. The last is necessary in the case third party contractors might use FOSS products. Haddad suggests a generic FOSS compliance process consisting of the steps of 1) scanning code, e.g., using FOSS scanning tools, 2) identification and resolution of flagged scenes, 3) architectural review, 4) linkage analysis, 5) legal review, and 6) final review. He also shows how to handle incoming FOSS compliance enquiries.

28. *Due diligence* is a term used for a number of concepts involving either an investigation of a business or person prior to signing a contract, or an act with a certain standard of care. Source: Wikipedia http://en.wikipedia.org/wiki/Due_diligence; accessed February 17, 2011.

4.4 FOSS Distributions

The term *distribution* is used for software that is bundled together in a collection, for example containing an operating system, the system setup, and a selection of application software. Collections of software are often selected with a specific theme or application area in mind, such as a desktop system, server, laptop, medical, educational, etc. Software bundles are considered as collective work. Bundling software requires that the licensing terms of the software in one bundle are compatible. Note that for such collections the GPL does not require other software to be free, as long as not linked or otherwise included in conflict with the GPL. Thus, both commercial software and software under the GPL can be included in one bundle, as long as the distributor has the permissions for the commercial software.

Several hundreds Linux distributions with an emphasis on different subjects are available²⁹. According to dictionaries, the term *distribution* refers to the commercial activity of transporting and selling goods from a producer to a consumer.³⁰ The Jargon File (Raymond, 1996; Steele and Raymond, 2000, version 4.2.3) gives the following definition:

A software source tree packaged for distribution; but see *kit*. Since about 1996 unqualified use of this term often implies 'Linux distribution'. The short for **distro** is often used for this sense.

Soon after the CD-ROM was available as a distribution medium, the *Prime Time Software for Unix* (Morin, 1993) and *PowerTools for Unix* (Peek et al., 1993) were released as distributions of software. These can be seen as early attempts to create a distribution of useful software focussing on a loose collection of tools as add-on to the operating system. A book or booklet, and specific installation programs followed with the CD-ROMs.

Today, an operating system comes packaged together with a selection of useful software. Examples for such distributions include Ubuntu, Debian, Red Hat, S.u.S.E, or several flavours of BSD. In many cases so-called live-CDs are available, making it possible to boot and run the software without installing it; lately also memory sticks can be used for booting distributions.

Early Linux-distributions were available on CD-ROM since the Internet did not have a significant penetration with enough bandwidth outside academic institutions. The distributions SLS and Slackware could be purchased by the price of a handling fee. Other distributions focused on specific goals; *Gentoo* is very customisable due to the *Portage-Technology*³¹; *SuSE*³² and *Red Hat* are commercial distributions, adding commercial software; *Fedora* is a free version of Red Hat; while *Debian* is governed by a social contract of its developers. From Debian several other distributions were derived, the most relevant being *Knoppix* and *Ubuntu*.

29. See <http://www.distrowatch.com>; accessed September 24, 2010.

30. There are other meanings of the word *distribution* which are not relevant here.

31. See www.gentoo.org; accessed September 24, 2010.

32. SUSE now is part of Novell.

EULA. Distributions are created for end users. Therefore a distribution contains an *End User License Agreement* (EULA), which states the licenses for the distribution, and its components. Since a distribution consists of many sub-components, several licenses might apply. The EULA states these different licenses and their terms, as well as disclaimers, and other terms of use. Note that a license, such as the GNU GPL, is not an EULA, because the copyright regulations enforce no constraints on solely running the software.

Packet Manager. Today, maintenance of the software usually is done by network-updates using the suitable packet management system. Technically, in addition to the operating system, and the software, a distribution needs a packet management system which is used for packaging, installation, update and maintenance of the entire system. For Linux distributions, two major packet management systems are predominant: *rpm*³³ for Red Hat, Fedora, SUSE, Mandriva, etc; and *dpkg* and *apt* using the *deb* format for Debian, Ubuntu, Knoppix, etc. Note that other packet management systems are available, e.g., Portage for Gentoo.

Distribution Planning. The major distributions tailored for the end-user market face requirements similar to commercial products. If these requirements are not met, history has shown that this distribution will be less used, and other distributions will be used instead. Therefore, distribution planning, release management, packet management system, and logistics around a distribution are essential.

Michlmayr (2007, 2009) and Michlmayr et al. (2007) point out that *release management*, i.e., the process when and how often to release new versions is a major part of distribution planning.³⁴ Since a distribution contains a large number of software packages, the major software packages of a distribution need to have a more or less synchronised release management, in order to be able to keep a distribution consistent. A distribution lives from presenting fresh software, such as new releases of the graphical user interface (e.g., KDE, Gnome). Therefore, the release dates need to be aligned. A regular release plan of software, both single software packages and distributions, is becoming more and more common.

4.4.1 Debian

Debian was first announced on 16 August 1993 by Ian Murdock, as a reaction to the perceived poor maintenance of the then dominant SLS Linux distribution. He released the Debian Manifesto, to ensure a non-commercial Linux distribution. In 1996, the first version of Debian was released; Bruce Perens took over as project leader, and Ean Schuessler suggested that Debian should establish a social contract with its users (Perens and Debian, 1997). Establishing a social contract was in contrast to the commercial Linux distributions (Perens, 1999).

33. See www.rpm.org; accessed September 24, 2010.

34. See also www.nuug.no/aktiviteter/20070508-reلمان-freeprogproj; accessed September 24, 2010.

The Debian project is founded on three documents: 1) the *Debian Social Contract* defines a set of basic principles by which the project and its developers conduct affairs; 2) the *Debian Free Software Guide* which defines criteria for software permissible in the distribution; and 3) the *Debian Constitution* which describes the organisational structure of the project.

4.4.2 Commercial Distributions

Commercial distributions, such as Red Hat and SuSE, base their software on FOSS. However, they may add commercial products with commercial licenses. These products may include specific drivers, application software (e.g., niche-oriented such as film editing, accounting, virtualisation), software that handles patented data format codecs, etc. Therefore, some of the commercial distributions can offer better support for some purposes.

The different licenses are usually referred to in the EULA. As long as the terms of the licenses of the single products are not breached³⁵, it is legally allowed to present a mix of FOSS and commercial software in a distribution. The use of the LGPL for libraries is essential for being able to include software libraries.

There are some discussions whether mixing commercial software and FOSS in one distribution is a good thing. On one side, the end user gets a service that would not be available using only FOSS, while on the other side, there are less incentives to keep software free, or use a FOSS alternative.

The commercial distributions often bundle their FOSS with several types of subscriptions, offering support, and additional software in a premium version that needs to be purchased. Lately, various cloud services using a freemium-model are easily accessible in some distributions, such as Ubuntu-One for Ubuntu. More on business-models suitable for free software can be found in Section 4.7.

Around a commercial distribution, often a company has interests while the efforts of a community are less predominant. In order to avoid the loss of a supporting community which is important for the further development, some distributions have split their efforts into a commercial distribution, and a free distribution. In this spirit, Red Hat has created Fedora as a free distribution.

4.5 Development Process

According to Benkler (2002) the production of goods such as software is based upon three different models; 1) managerial command systems like firms or organisations, where hierarchies define the line of command; 2) markets, where the concept of transaction costs define the production; and 3) peer production, where other incentives govern than in the other two models, and which is based on decentralised information gathering and exchange. We have already discussed the Commons Based Peer Production (CBPP) and its characteristics in Chapters 2 and 3.

35. It is not allowed to include a GPLed software library that links to a commercial programme in a distribution.

FOSS can follow the CBPP paradigm, or be governed by other development principles, such as one developer coding the entire code base of a product. However, software products that are alive, and have a community gathered around the code base, often use CBPP or related methods. The development process of FOSS following CBPP is different from the development process of projects following other paradigms. FOSS can follow different development processes depending on the type of the project, independently of the license. However, giving access to the source code gives users the opportunity to give feedback and code suggestions to the developers.

The GNU-Emacs is an example of a FOSS project that has been driven as a managerial command system, with RMS as project architect and as project manager.³⁶ This gives other developers only a limited influence. Consequences can be a project split³⁷, as Raymond (1999) remarks. Linux is an example of projects that use CBPP. These projects are managed differently; However, there is the need for a different structure so that projects can scale. Successful CBPP-based FOSS projects need a suitable communication infrastructure, including source code repository and wiki. Another characteristic is that the granularity of tasks, that is that the tasks have a limited size and complexity so that these can be managed independently. Additionally, the general principles for successful CBPP projects, as outlined in Section 3.3, need to be obeyed.

In FOSS projects the project owners are responsible for coordination of many volunteers, maintain a database, take decisions, perform some of the programming, etc. Users of FOSS applications can suggest changes in the software, and thus contribute to further development. For most of the projects web pages are available, including possibilities for download, information on the project, communication between developers, addresses for reporting bugs, etc.

There are sites that host many projects as an infrastructure. These include, e.g., Sourceforge³⁸, Savannah³⁹, gitorious⁴⁰, github⁴¹, or CodePlex⁴². The distribution model for FOSS is often based on web servers with download-facilities, and the Internet as an infrastructure. Note that some of these sites might have restrictions on the applicable FOSS licenses, and the nature of the product.

36. It appears that GNU Emacs now is developed by a team as a look at the commiter list for the Emacs code repository shows. See <http://git.savannah.gnu.org/cgi/emacs.git>; accessed August 30, 2011. Many contributors have commit-access. Also, RMS handed over the Emacs project leader position some years ago. See <http://lists.gnu.org/archive/html/emacs-devel/2008-02/msg02140.html>; accessed August 30, 2011. In the same communication, RMS comments that: *This is the fourth time that the Maintainer of GNU Emacs has been someone other than me. Previous maintainers include Joe Arceneaux, Jim Blandy, and Gerd Moellmann.* We thank Håkon Stordahl for commenting this.

37. Examples of this are the *GNU Emacs* and *XEmacs* split; or the *NetBSD* and *OpenBSD* split; or the *OpenOffice* and *LibreOffice* split.

38. See <http://www.sourceforge.net>; accessed September 24, 2010.

39. See <http://savannah.gnu.org>; accessed September 24, 2010.

40. See gitorious.org; accessed October 1, 2010.

41. See github.com; accessed October 1, 2010.

42. See <http://en.wikipedia.org/wiki/CodePlex>; accessed August 25, 2011.

4.6 Costs of FOSS

Even though FOSS is available for everybody to access, view and use, it is not necessarily without costs. Note that there may be differences in the cost structure for personal use and for use in an enterprise. For users, the costs of FOSS arise in accessing the software, installation, maintenance, royalties for licenses, building the necessary infrastructure, implementation of additional functionality, and so on.

The *total cost of ownership* (TCO) includes all the above mentioned costs, and provides an economic metric to compare different FOSS-based systems with commercial alternatives. TCO is a management accounting concept with the purpose to help consumers and enterprise managers determine direct and indirect costs of a product or system. All software has a TCO which includes the sale price, any hardware and software upgrades, maintenance and technical support, and training. Note that some costs, such as time and frustration may be hard to measure. These components of TCO should be part of the decision to use any software: *a*) price, *b*) opportunity costs, and *c*) other costs.

Perry and Gillen (2007) discuss a seven-step process for understanding, measuring, and articulating the business value of an IT-project. When establishing the TCO, they divide the costs into six different categories based on 300 interviews:⁴³ 1) staffing (69%); 2) downtime – user productivity (15%); 3) IT staff training (8%); 4) server hardware (7%); 5) software (7%); and 6) outsourced costs (3%). Note, however, that the impact of these categories differs between type of projects, enterprise, purpose, etc. Therefore, exact numbers that are valid in general are difficult to establish.⁴⁴

Open source proponents claim that even if open source requires more expertise, the TCO is ultimately lower. Proponents of the commercial business model claim that the required expertise is daunting and other costs of proprietary solutions are exaggerated.⁴⁵ This is illustrated in Figure 4.2.

There have been long-lasting discussions whether open source products or commercial products are better, and which of these has a lower TCO. Not surprisingly, it is easy to find for each product a report and calculation that positively supports this product. Wheeler (2007) lists in Chapter 7 of his document many examples that support either open source or commercial products with respect to lower TCO.⁴⁶ Wheeler also argues apart from TCO that FOSS is worth using. We also refer to another document that shows the positive sides of FOSS regarding the TCO (acronym SA).

43. The percentage numbers in parentheses denote the the division for this item as presented by Perry and Gillen (2007).

44. The whitepaper by Perry and Gillen (2007) was performed by the International Data Corporation (IDC) (see <http://www.idc.com>; accessed February 17, 2012), and sponsored by Microsoft. The data presented in this whitepaper were used by Microsoft to argue that open source products like Linux have a higher TCO than corresponding Microsoft products.

45. See http://www.netc.org/openoptions/pros_cons/tco.html; accessed August 7, 2011.

46. David A. Wheeler appears as a supporter to the open source movement.

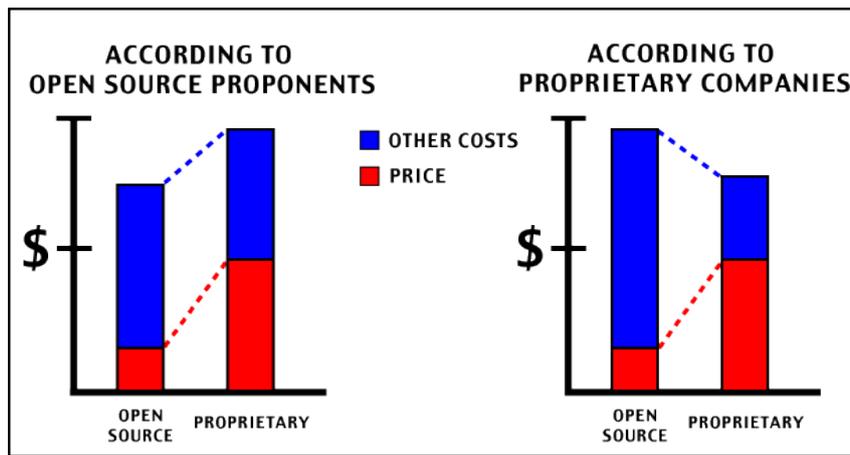


Figure 4.2. Open source proponents and proprietary companies disagree on the total cost of ownership. Developed by the Northwest Regional Educational Laboratory, Portland, Oregon.

4.7 FOSS Business Models

Despite FOSS being available at little cost, and possible to copy at low cost, the FOSS industry is a growing multi-billion Euro industry. The exact number is difficult to verify since many of the enterprises get their income from additional sources beyond FOSS. Companies like Red Hat, SuSE (Novell), SugarCRM or Canonical, and associations, like the Mozilla Foundation or the Apache Foundation, fill important shares of the market for products in information technologies. Large enterprises, such as IBM, Google, Nokia, Apple or Oracle use FOSS actively in their portfolio, but do not base their business models solely on FOSS. It is evident that commercial business models that are based on the sale of licenses are not viable. For most enterprises, FOSS business models are based on some kind of cross-subsidisation.

A business model describes the rationale of how an organisation creates, delivers, and captures value in the form of economic, social, or other metrics.⁴⁷ The process of developing a business model is part of a business strategy. Osterwalder (2004) presents an ontology of business models. A large variety of business models has been developed over time, such as *razor and blades*, *bricks and clicks*, *collective*, *cutting out the middleman*, *direct sales*, *franchise*, *fee in – free out*, *monopolistic*, *premium*, and so on. Since general business models are beyond the scope of this chapter, we refer to the survey by Zott et al. (2010) for further reading.

Anderson (2009) points out that the costs for production and distribution of FOSS are very low, and tend to converge towards zero. This is the starting point of what he calls the *bits economy* where goods can be obtained for free. However, in order to create a substantial industry, business models need to be in place. In the following, we present business models relevant to FOSS.

47. See http://en.wikipedia.org/wiki/Business_model; accessed January 14, 2012.

Cost Reduction. FOSS can be used to reduce costs in enterprises and public entities, also in those that are not directly involved in software development. FOSS can be used for supporting services such as accounting, web profile, public relations activities, and so on. Here, the discussion on TCO as outlined in Section 4.6 comes into the picture. Note that these enterprises often contribute to the development of the FOSS products they use.

In the cost reduction model, it is essential that the main revenue of the enterprise does not come from the sale of the software in question. Thus, this software is more a vehicle to increase the main source of income, such as sale of non-software products.

It is said that Sun Microsystems had cost reductions due to reduced license payments in mind when they acquired StarDivision in 1999, as well as trying to reduce the market shares of their competitor Microsoft. At that time, the development costs of OpenOffice appeared to be less than the license payments for Microsoft Office products.

Services. Many enterprises offer services that are based on FOSS. These services might include hosting of files; hosting and serving web content; music, images and video content; services for accounting; text processing; and services in the cloud, including the layers infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Note that the GNU Affero License addresses issues arising from SaaS, as outlined in Section 4.8. Often, a freemium business model is used.

Support and Consulting. Even though FOSS is available for free, enterprises can charge for support, maintenance, and consulting. Typically such services are offered for customisations to the specific needs of a customer. Revenue can also come from education and courses on FOSS and specific products.

Enterprises that successfully implement this business model often distribute software without charging for it. Many of these enterprises derive revenues from support services and training for corporate customers who run mission-critical business systems on this platform. Canonical Ltd., who developed the *Ubuntu* distribution, represent this business model. There are several examples for enterprises that sell support and consulting services, independent from a specific product that they own, such as Linpro or Freecode⁴⁸

Direct Cross-subsidisation (Loss-Leader). A *loss leader*, or simply a leader, is a product sold at a low price, often below cost or without charges, to stimulate other profitable sales.⁴⁹ According to Anderson (2009), this model has been used for over a century for traditional products. Often, the free or subsidised product can only be used together with another product that needs to be purchased. In other cases, the free product is used for promotion purposes only. Another variant is the *buy two – get one free* model. Examples for this business model related to FOSS includes IBM who distribute the software prod-

48. See linpro.no, resp. freecode.no; accessed March 5, 2012.

49. See http://en.wikipedia.org/wiki/Loss_leader; accessed January 15, 2012.

uct *Eclipse* in order to strengthen their brand, and to attract paying customers for their other products.

Freemium. In the freemium model⁵⁰ an enterprise offers a range of products or services for free while improved or extended versions can be purchased for a price. Generally, only a small fraction of the customers needs to sign up for the premium version in order to create a sustainable business model; typically, less than 5% of paying customers subsidise the costs of both the free and the premium version (Anderson, 2009). The term *freemium* has apparently been defined in 2006 by Fred Wilson and Jarid Lukin in a blog discussion. Wilson defines freemium as:⁵¹

Give your service away for free, possibly ad supported but maybe not, acquire a lot of customers very efficiently through word of mouth, referral networks, organic search marketing, etc., then offer premium-priced value added services or an enhanced version of your service to your customer base.

The freemium model is not only applicable to FOSS, but to all kinds of products. A variant of the freemium model is used by enterprises who offer price-reduced early versions of a product for users who in turn provide feedback that will be incorporated in the premium version of this product. Enterprises using the freemium model can reduce their cost for marketing, development and testing, which they also tend to have a high innovation rate.

The freemium model can only work if the customer receives an added value for the price he or she pays. This can either be achieved by closing down parts of the source code, or by offering goods or the use of an infrastructure that comes with a price.

The freemium model works well for services that are based on FOSS. Here, an enterprise offers services based on FOSS on an infrastructure that they own and maintain. Often, hosting services and software as a service targeting certain applications are candidates for the freemium model. Examples include text processing, accounting, and hosting-services where certain functionality and volumes up to a certain amount are free, while extended functionality and larger volumes are paid services. In some cases enterprises offer dual-licensed products to users for free, often with terms for non-commercial use, while commercial users need to pay a price to use this software in their products or services.

Dual Licensing. Many FOSS enterprises present their business model to be *dual licensing*. However, dual licensing is considered more a tool to support other business models than a separate business model. Business models building on dual licensing include freemium, and open core. Meeker (2005) outlines how the business models using dual-licensing work.

50. freemium = free + premium.

51. See http://avc.blogs.com/a_vc/2006/03/my_favorite_bus.html; accessed January 15, 2012.

Open Core. Open Core⁵² is a practice, where companies use a combination of dual-licensing, closing parts of the source code, and the *freemium* business model. In this case, the company often maintains a dual-licensed software that represents some type of middleware that opens up for extensions, such as plug-ins. The open core principle works only for permissive licenses, or when copyright assignment is used.

The GPL together with copyright assignment can be used to effectively prevent others from competition: Contributors to software using open core can choose only to release under the GPL, in which case the company behind the product will not include the change. However, when assigning copyrights, the contributor has the same rights as any other GPL licensor while the company can use this code both for GPL and proprietary licenses. This effect is not intended by the design of the GPL.

Packaging and Simplification. Enterprises that use this model typically develop installers around complex software that requires many advanced settings for installation. While the end-user is free to install and configure the software manually, the revenue comes from selling the specially tailored installer. However, many of the enterprises using this model have failed to scale economically. Another threat to this model may come from distribution developers or communities who provide such installers as FOSS.

Hardware. Software in hardware appliances, such as routers, TVs, smart-phones, net-pads, diverse machines, etc. are often based on FOSS. Using FOSS instead of developing a specially tailored (operating) system reduces development costs drastically since much of the development is done by a FOSS community. Additionally, the enterprise may receive feedback from a community and the users, as well as fixes, new ideas, and so on.

Widget-frosting. Enterprises that sell hardware provide drivers as FOSS. Thus, hardware drivers can be adapted to other target systems by a community. While an enterprise may not get a revenue directly from selling the driver software, this contributes to making the hardware available for many platforms without increasing the development costs. In many cases, the drivers and application software can be downloaded from supporting web sites both in binary form and as source code. Widget-frosting is used for hardware such as printers, scanners, USB- and Ethernet controllers, storage controllers, etc. to support Linux users.

Note that some manufacturers only release a binary version of their drivers at no cost. These can be linked into the operating system or into the system software using a defined interface. In this case, the software is not available as FOSS, but the hardware can be used in FOSS-based systems.

52. See http://en.wikipedia.org/wiki/Open_core; accessed December 22, 2011.

Accessorising. This business model is often used by foundations or businesses to promote FOSS and to support communities. They offer accessories such as coffee mugs, T-shirts, manuals, books, compatible hardware, systems with FOSS pre-installed, and other related accessories. The publisher O'Reilly Associates, Canonical, and many of the well-known FOSS communities follow this model.

Advertising and Search Forwarding. Some enterprises and foundations offer space for promotion, advertisements, and forwarding of searches. As an example, the Mozilla foundation gets a large part of their income from search engines, such as Google. By forwarding searches in the Firefox web browser to a specific search engine, the Mozilla foundation gets a revenue from the search engine owner.⁵³

Donation-based Model for Non-profit Foundations. Foundations, such as the Mozilla foundation, the KDE foundation, the FreeBSD foundation, or the Software Conservatory, allow users and organisations to offer donations. These donations are used to fund and extend community activities.⁵⁴

FOSS Business Models in Developing Countries. Developing and transition countries have often a social and societal profiles where local ownership, local value addition, empowerment and participation can be beneficial. FOSS business models can be suited to sustain this, as proposed by InWent (2011). In another document, InWent (2010) presents teaching material for FOSS business models in Africa.

4.8 Characteristics of FOSS

As discussed previously in this chapter, the openness of FOSS provides the user with many advantages. These include that the user can inspect the code, consider its quality, performance, security issues, fault possibilities, etc. The user can also adjust the code as needed, fix bugs, etc. However, these advantages of FOSS require an active user who has skills within computer software design and implementation. As an alternative, the user can ask a person with computer-skills for help, possibly as a paid service.

This fact may be one of the reasons that FOSS is today used by enterprises who can afford to purchase the required services, as well as by people with computer-skills. Among other groups there is a widespread sceptical attitude towards FOSS.

For many users, the commercial software is what they expect when it comes to the use of computers. This expectation ranges from functionality, via look-and-feel, to the way how malfunctions are handled, e.g., by warranties. Any change of this will confuse the inexperienced user, and thus drive her or him away from FOSS. Possible FUD⁵⁵ campaigns

53. This deal has caught much attention lately, since it has been re-negotiated in December 2011; see <http://blog.mozilla.com/blog/2011/12/20/>; accessed February 23, 2012.

54. Also foundations offering open content, such as Wikipedia, use the donation-based model.

55. Fear, uncertainty, and doubt (FUD) is a tactic of rhetoric and fallacy used in sales, marketing, public

from commercially working competitors contribute additionally to the reluctance to use FOSS.

In most cases, software is run for a purpose, that is the user expects a certain service, performance and functionality in a given environment. For some applications there are no FOSS replacements (yet) available. For example, an accounting system in an enterprise might be tailored to specific needs built on commercially available building blocks. Replacing such a system by a FOSS products is often not viable, since certain APIs or scripts need to be implemented. Usually, a replacement of such a system is a major undertaking.⁵⁶

Even though FOSS products implement similar functionalities, it is not always possible to use FOSS products. As an example we mention OpenOffice which implements an office suite in line with Microsoft Office. In fact, many features of OpenOffice re-implement Microsoft Office, and it is said that Sun Microsystems purchased StarOffice to implement an office suite that does not require licensing fees, and that the costs of the development of OpenOffice are less than the licensing fees for a similar installation.⁵⁷ However, for some documents, e.g. legal documents, it is important that these are exactly presented as the system that generated these. Even though importing and exporting functionality exist, the transformation does not always produce identical results, and information may get lost in that process.

In order to overcome some of the problems not being able to run FOSS products, the users can utilise a mixed environment, running software in some type of virtual machine or an emulator. Examples include wine, kvm, VMWare, QEMU, etc. Note that some of these are FOSS, others are commercial products. Often, administrative products in businesses are proprietary, and need to be run on virtual machines, or on external machines with graphical windows on the client machine. Nonetheless, virtual machines enable these mixed environments. Note also that some distributions of Linux offer both FOSS and commercial software, such as SuSE, RedHat, etc.

Interplay with proprietary systems. Proprietary systems often employ proprietary protocols or other proprietary standards, e.g., for storage of information. These standards are unknown to outsiders, and are often protected by copyright, patents, or contracts. In practice many of these protocols and formats are not available to implementors of FOSS. Note that in some cases it is even not legal to implement certain standards with FOSS, since the licensing terms do not allow this.⁵⁸

On the other hand, since a substantial part of the users uses FOSS-based systems or systems developed by another provider, software providers need to open up in order not

relations. See http://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt; accessed August 10, 2010.

56. See Rahemipour (2010); Rahemipour and Effenberger (2010) for practical recommendations how OpenOffice can be adjusted to fit into a business environment.

57. Note that Sun Microsystems was acquired by Oracle in 2009.

58. An example is the encoding of mp3 files. The licensing terms require that a fee is paid per encoder, and per encoded file. These terms cannot be fulfilled using FOSS.

to loose market shares. The use of a proprietary protocol where no implementations for other platforms exists, will only work as long as some kind of monopoly situation can be maintained. As soon as users of another platform arrive, or interoperability is required, implementations on other platforms will occur. In many cases, an enterprise using the proprietary software model will license their knowledge under commercial conditions. However, for open source platforms, such licensing usually will not work, and competing implementations, often derived from some kind of reverse-engineering, will occur that are more or less interoperable. Since this could lead to frustrated users, the acceptance of a proprietary system with closed protocols could decrease. In some cases, open source communities have developed competing, better products that eventually replaced the proprietary technology.

Examples of technologies that have been developed as a reaction to proprietary technologies include (a) mono, a cross platform, open source .NET development framework⁵⁹; (b) Moonlight⁶⁰, which is an open source implementation of Silverlight; (c) OpenOffice and LibreOffice, an office suite⁶¹ that replaces Microsoft Office; (d) The Gimp⁶² as a replacement of Photoshop; and so on.

For some products, compatibility can be achieved by the use of converters that transform the input- or output-formats of proprietary systems to the respective formats of their open source counterparts. However, these are not always available, such as for Microsoft Visio⁶³ or for certain mindmapping software. In these cases, the end user will suffer by not being able to re-use own work, or being able to access others work.

For FOSS to work properly, the use of open standards⁶⁴ that are publicly available is recommended. If these are not available, the use of other protocols that follow the definition of *open* is preferred. Note also that some standards that are open, such as some standards for multimedia technologies released by the ISO, can contain patents, which can be a problem for implementing such functionality as FOSS.

Hardware issues. In order to be useful, hardware needs to be supported by the operating system, system drivers, or the application software. When new hardware comes on the market, the FOSS community might not have had the opportunity to implement the drivers necessary to use the hardware. This is the case especially when the specifications of the hardware are not obtainable.⁶⁵ Some manufacturers only release a binary version of their drivers at no cost. These can be linked into the operating system or system software using a defined interface. In this case, the software is not available as FOSS, but

59. See www.mono-project.com/; accessed August 24, 2011.

60. See www.mono-project.com/Moonlight; accessed August 24, 2011.

61. See www.openoffice.org/; accessed August 24, 2011 and www.libreoffice.org/; accessed August 24, 2011.

62. See www.gimp.org/; accessed August 24, 2011.

63. See en.wikipedia.org/wiki/Microsoft_Visio; accessed August 24, 2011.

64. See en.wikipedia.org/wiki/Open_standard; accessed August 25, 2011.

65. Sometimes the opposite happens. Lately, the openly available USB 3.0 specification has been implemented for Linux as the first operating system having the necessary drivers in place.

can be used together with FOSS-based systems. Other vendors publish drivers to their hardware as FOSS in order to get better software support, and to make use of innovation in the form of better software from the community around the product.

Experience shows that it usually takes some time before hardware products are fully supported by FOSS-based products. Therefore, buyers of hardware often need to do research whether a product is compatible with the FOSS they are using. For commercial products, drivers often come with the hardware, attached on a CD-ROM or obtainable from the Internet. In some cases, e.g., for some 3G communication devices, only suboptimal implementations are available for the Linux operating system. However, for the majority of hardware, especially hardware that has been available for some time on the market, the support of FOSS is excellent, as long as there are no other obstacles, such as patents.

Universal design. Universal design refers to a broad spectrum of products and environments that are usable and effective for everyone.⁶⁶ It emerged from *barrier-free design* and *assistive technology*, and recognises the importance of how things are perceived in the minds of all users. In principle, FOSS should be ideal to produce universally designed software. However, in practice there are several obstacles to overcome, such as missing or insufficient hardware support for devices necessary to give access to special target groups, lack of interest or ignorance of community members in FOSS projects, or insufficient API design for universally designed features. Also the ignorance of most users requiring universally designed products towards FOSS might play a role, since CBPP by these users could foster a better commons, namely universally designed products that fit better their needs.

Functionality and usability are most important for all target groups using software. Implementing universally designed software requires a certain architectural design, access to APIs, and access to protocols and interfaces to use both hardware and software products. Several distributions offer functionality for universal design, as well as for internationalisation, which can be switched on if desired. However, currently the support is quite limited, and available for some larger target groups only, such as visually impaired or hearing impaired.

Often, there is only limited support for aids that are provided by the governmental organisations⁶⁷. While some technically skilled users have managed to develop the necessary interfaces for their needs, most users need to give up on using software other than the commercially supported software. In some cases, the development of FOSS is not possible due to patents or business secrets that might be part of the interfaces or protocols.

In a presentation in 2007, Klaus Knopper, the developer of the Linux distribution *Knoppix*⁶⁸ shows examples of application- and desktop helpers, OpenOffice Accessibility, and

66. See http://en.wikipedia.org/wiki/Universal_design; accessed August 15, 2010.

67. In Norway, NAV and the *hjelpemiddelsentralen* <http://www.nav.no/hjelpemiddelsentralene>; accessed September 24, 2010, are responsible for supporting aids for target groups.

68. See <http://www.nuug.no/aktiviteter/20071211-accessibility/>; accessed September 24, 2010.

the *ORCA screenreader*; he remarks that making information *accessible* is not as easy as often advertised by software vendors taking into account the various different capabilities and possibilities of users with and without disabilities.

Software as a Service. *Software as a service* (SaaS) is a software delivery model in which software and its associated data are hosted centrally on servers, while the results of running this software are available on the user's terminal. The term SaaS is often connected to running services in the Internet cloud, and is a further development from the Service Oriented Architecture (SOA) which is extended from web services to applications, platforms and infrastructures (Tietz et al., 2011). Often, the layers in the Internet cloud are denoted as 1) *Infrastructure as a Service* (IaaS), 2) *Platform as a Service* (PaaS), and 3) *Software as a Service* (SaaS), with a management layer for all three of these layers.

While practically every Internet service is driven by some underlying software running on a server, the term SaaS is often used in the context of business applications⁶⁹, and for applications for personal computing, such as document processing, spread sheets, presentations, or image processing. The latter category is often combined with offers from the providers to host content. As for the first category, SaaS has become a common delivery model for most business applications, including accounting, collaboration, customer relationship management (CRM), enterprise resource planning (ERP), invoicing, human resource management (HRM), content management (CM) and service desk management.

SaaS can offer a variety of advantages, such as low costs for the users of standard services, outsourcing of infrastructure and maintenance to the provider, high scalability, rich functionality, and so on. As business models we find subscription, pay for use, freemium, or even free access, often as an advertisement-based model.

Stallman (2010) argues that the use of SaaS is a challenge to the software freedoms proclaimed by the FSF.⁷⁰ In his opinion, the use of SaaS lets the user loose control since the data are processed on a server that is running some software under someone else's control. The threat of spyware running on the SaaS servers, e.g., for advertisement purposes, or other malicious activities, such as altering content, cannot be neglected. Therefore, the use of free software that must be identical with the software running on the SaaS server can be one step in the right direction, so that the users can check how their data are processed.

The GNU Affero General Public License is a modified version of the ordinary GNU GPL version 3 with one added requirement: if you run the program on a server and let other users communicate with it there, your server must also allow them to download the source code corresponding to the program that it's running.⁷¹ If what is running there is your modified version of the program, the server's users must get the source code as you

69. See http://en.wikipedia.org/wiki/Software_as_a_service; accessed August 7, 2011.

70. See <http://www.gnu.org/philosophy/who-does-that-server-really-serve.html>; accessed August 7, 2011.

71. See <http://www.gnu.org/licenses/why-affero-gpl.html>; accessed August 7, 2011.

modified it. While the GNU Affero GPL affects developers of free programs that are often used on servers, the problem of controlling what really is going on with the customer's data on a server cannot be solved with licensing alone.

Both the ordinary GNU GPL, version 3, and the GNU Affero GPL allow users to link together modules under these two licenses in one program.

While SaaS providers technically are running the service and its data entirely on a server, including processing and hosting of user data, SaaS must not be confused with running a service on a local computer, typically in a browser, with software provided from an external server. Often SaaS services are enriched with this kind of software in the form of JavaScript programs or Java applets which might be proprietary software. When accessing a service, these are often run without informing the user. Note that SaaS and proprietary software in the browser often are combined.

According to Stallman, SaaS and proprietary software lead to similar harmful results, but the causal mechanisms are different. While using free software in a server gives the software freedoms to the provider, it does not protect the end users from the effects of SaaS of losing control.

Patents. Patents are temporarily limited rights to exploit a genuine idea exclusively. Rights holders can set the terms how others can use technologies covered by the patent, e.g., by paying a fee. However, in some occasions, patents can prevent FOSS from being used, as the example of the patent covering the mp3 technology shows.

According to Bruce Perens⁷², *software patenting is generally hostile to FOSS, because patent holders require a royalty payment that isn't possible for developers who distribute their software at no charge. There are also many other reasons that the software patenting system is broken and actually works to discourage innovation.*

The mp3 technology covers en- and decoding sound files, such as music efficiently. The technology is patented by Fraunhofer IIS and Thomson Consumer Electronics. The terms require that a fee is paid to the patent holders for each sold version of the encoding software, as well as a fee for each encoded file. There have been attempts to implement the mp3 encoder as FOSS: *bladeenc*. However, *bladeenc* cannot be used legally in most countries.⁷³ As a reaction, *Ogg Vorbis* was developed as a replacement. *Ogg Vorbis* has similar specifications as mp3, but is not based on patented technology. *Ogg Vorbis* has reached a relatively good penetration in the market.⁷⁴

Other obstacles for the implementation of multimedia FOSS include patents and closed media formats where reverse engineering is prohibited. Sometimes altered Win32-libraries are used, but also this can cause licensing problems. The use of media formats is, however, more an issue of open standards, which we treat in Chapter 7.

72. See <http://perens.com/policy/open-source/>; accessed August 28, 2011.

73. See <http://www2.arnes.si/~mmilut/>; accessed September 24, 2010.

74. *Ogg Vorbis* is used extensively in games.

Perens points out⁷⁵ that *proprietary file formats and intercommunication protocols are used by a software manufacturer to lock out the products of other manufacturers and open source. Perens believes that business and government should insist on publicly documented file formats and intercommunication protocols that require no royalty or discriminatory licensing. There is sufficient space for a business to differentiate their product in all of the other parts of the program that are not concerned with the technical implementation of file formats and intercommunication.*

The GNU General Public License version 3 (GPLv3)⁷⁶ addresses patent issues explicitly in order to counter the threats to the software freedoms from a more and more pervasive enforcement of patent rights. In the GPLv3, the licensor must give the permission to use all patents under the ownership or control of the licensor. As a consequence the GPLv3-covered software can be used without worrying that a desperate contributor will try to sue them for patent infringements later. Note that if a licensee tries to use a patent suit to stop another user from exercising those rights, the license will be terminated.

Digital Rights Management. Digital Rights Management (DRM) protects content from unauthorised access (Abie, 2007). However, DRM is a technology that potentially imposes restrictions on software freedom and on access to other copyrighted work. According to the FSF, the DRM technology, which they name *Digital Restrictions Management*⁷⁷, is a threat to the software freedoms. In the GNU General Public License version 3 (GPLv3)⁷⁸ it is stated that compatibility to the GPLv3 is only achieved when *No covered work constitutes part of an effective technological protection measure*. While this sounds like a general prohibition to implement DRM systems, the licenses FAQ⁷⁹ explains that releasing code to develop any kind of DRM technology is allowed and will not count as an effective technological protection measure. This implies that the DMCA (Digital Millennium Copyright Act), the European Union Copyright Directive, and similar laws cannot be applied if someone breaks⁸⁰ a method implemented under the GPLv3. Effectively, the DRM clause is designed to avoid restrictions of software imposed by these law.

Security Issues. For over a decade there has been a dispute whether FOSS is more secure than commercial alternative. Hansen et al. (2002) claim that *open and co-operative software development can lead to robust and reliable products, but that adequate diligence during the entire development process and during the evaluation by experts* is needed. Here, by security we mean the absence of vulnerabilities that could be exploited so that damage, cost, or unavailable services could occur. Using this definition, security is closely tied to software

75. See <http://perens.com/policy/open-source/>; accessed August 28, 2011.

76. See www.gnu.org/licenses/quick-guide-gplv3.html; accessed August 25, 2011.

77. See www.defectivebysdesign.org; accessed August 25, 2011.

78. See www.gnu.org/licenses/quick-guide-gplv3.html; accessed August 25, 2011.

79. See www.gnu.org/licenses/gpl-faq.html; accessed August 25, 2011.

80. In the past, there have been examples where (rather poorly designed) technical protection measures have been reverse-engineered and made available to others, such as the Content Scrambling System (CSS) for DVD content; see en.wikipedia.org/wiki/Content_Scramble_System; accessed August 25, 2011.

quality issues which are discussed in Chapter 5.

Wong (2002) identifies the reason for 90% of security vulnerabilities with buffer overflows, format string vulnerabilities, authentication, authorisation, and cryptography weaknesses. A report from the SANS Institute by Phillips (2003) lists buffer overflows, format string vulnerabilities, heap overflows, and issues with programming languages. All of these are in most cases caused by bad software quality. The question is whether FOSS or proprietary software is the better model to create better software. Phillips concludes that both sides have legitimate arguments for why their programming model is better, although the open source world has more compelling arguments.

The closed source developers build much of the security on the so-called “security through obscurity”, which means that possible attackers cannot know how a system is built and therefore need more resources. According to open-source proponents this argument is void, while closed-source proponents assume that attackers have an easy job find weaknesses in FOSS due to its openness. However, following the daily press about vulnerabilities, most often proprietary systems show vulnerabilities; only to some extent this can be explained by a larger market share.

In a recent study, Schryen (2011) looks at the question *Is Open Source Security a Myth?* He uses an empirical analysis by comparing 17 selected, widely used open-source and closed-source applications regarding vulnerabilities. The author retrieves the data from the MITRE database⁸¹ on common vulnerability and exposures, and the National Vulnerability Database⁸². He looks into several metrics, such as mean time between vulnerability disclosures, development of vulnerability disclosure over time and their severity, unpatched vulnerabilities and their severity. While a first analysis suggests that open source software is more secure than their closed source counterparts, a further statistical analysis shows that the differences are statistically not significant. Other properties, such as the patching behaviour, are dependent on the software vendor’s policy rather than the programming style.

Looking into reasons why open source software should be more secure, we find the quote by Linus Thorvalds: *Given enough eyeballs, all bugs are shallow*. Warfield (2003) suggests, that the ever-ongoing code-review will promote more secure coding techniques, also since the programmers in FOSS seek reputation in their community. He also argues why myths of lacking source control in FOSS, no one really looks at the source, and anyone could manipulate FOSS to the worse do not hold. Obasanjo (2002) argues that secure software is independent of programming model; instead, certain practices such as (a) formal methods; (b) code audits; (c) testing; (d) design reviews; and (e) codified best practices. While many FOSS communities use these practices, also proprietary developers follow these. Since this is independent of programming model, the benefit of FOSS is when making decisions whether a candidate software to be installed is secure due to its openness.⁸³

81. See cve.mitre.org/cve/; accessed November 25, 2011.

82. See nvd.nist.gov; accessed November 25, 2011.

83. Following this argument, the possibility to view the source code would be enough. Microsoft imple-

References

- Habtamu Abie. Frontiers of DRM knowledge and technology. *IJCSNS*, 7(1):216–231, 2007.
- acronym SA. Open source software vs. commercial software: Migration from windows to linux – an IT professional’s testimonial. article on the Internet, unknown year. URL members.apex-internet.com/sa/windowslinux. author uses acronym SA.
- Chris Anderson. *FREE: The future of radical price*. Random House, 2009. ISBN 9781905211470.
- Brian Behlendorf. Open source as a business strategy. In Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources – Voices from the Open Source Revolution*. O’Reilly, 1999.
- Yochai Benkler. Coase’s Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal*, 112:369–446, 2002.
- Douglas Coupland. *Microserfs*. Harper Perennial, June 1996. ISBN 0060987049.
- Chris DiBona, Sam Ockman, Mark Stone, Brian Behlendorf, Scott Bradner, Jim Hamerly, Kirk McKusick, Tim O’Reilly, Tom Paquin, Bruce Perens, Eric Raymond, Richard Stallman, Michael Tiemann, Linus Torvalds, Paul Vixie, Larry Wall, and Bob Young. *Open Sources: Voices from the Open Source Revolution*. O’Reilly Media, 1st edition, 1999. ISBN 1565925823.
- Jennifer Edstrom and Marlin Eller. *Barbarians Led by Bill Gates: Microsoft From The Inside: How The World’s Richest Corporation Wields Its Power*. Holt Paperbacks, reprint edition, June 1999. ISBN 0805057552.
- Joseph Feller and Brian Fitzgerald. A framework analysis of the open source software development paradigm. In *Proceedings of the 21st Annual International Conference on Information Systems (ICIS 2000)*, pages 58–69, Brisbane, Australia, 2000.
- Bill Gates. *Business at the Speed of Thought: Using a Digital Nervous System*. Penguin, February 2001. ISBN 0582343003.
- Bill Gates and Collins Hemingway. *Business at the Speed of Thought: Succeeding in the Digital Economy*. Penguin Books Ltd, May 2000. ISBN 0140283129.
- William H. Gates, Nathan Myhrvold, and Peter Rinearson. *Bill Gates*. Viking, 1995.
- Eleftherios Gkioulekas. *Developing software with GNU*. <http://www.amath.washington.edu/~lf/tutorials/autoconf/toolsmanual.dvi>, 1999. Note: several versions

mented such a mechanism by “Shared source” for its software, where software can be disclosed to business partners. See http://en.wikipedia.org/wiki/Shared_source; accessed August 25, 2011. Note that most of the shared source licenses are not considered free by the FSF; only two of these licenses are considered open by the OSI, as also outlined in Section 4.3.

- available on the Internet, e.g., <http://www.mars-attacks.org/~boklm/divers/doc/toolsmanual.pdf>.
- GNU. Various Licenses and Comments about Them. *Web pages*, <http://www.gnu.org/philosophy/license-list.html>, 2002.
- Ibrahim Haddad. Open-source compliance. *Linux J.*, 2009(185):5, 2009. ISSN 1075-3583.
- Marit Hansen, Kristian Köhntopp, and Andreas Pfitzmann. The open source approach opportunities and limitations with respect to security and privacy. *Computers & Security*, 21(5):461–471, 2002.
- Jordan Hubbard. *FreeBSD Handbook*, chapter A Brief History of FreeBSD. The FreeBSD Documentation Project, 1995-2010. URL <http://www.freebsd.org/doc/handbook/history.html>.
- InWEnt. ict@innovation: Free your it-business in africa! advanced training material on african free and open source software (FOSS) business models for IT-SMEs. Technical report, ict@innovation, July, 6 2010.
- InWEnt. Business models in free and open source software in developing countries. web publication, 2011. URL http://www.it-inwent.org/e2484/e3407/index_eng.html.
- Donald E. Knuth. *The TeXbook*. Addison-Wesley Professional, spi edition, January 1984. ISBN 0201134489.
- Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, January 1986. ISBN 0201134446.
- Heather Meeker. Dual-licensing open source business models – mixing proprietary terms with the gpl. *SYS-CON Media*, April 6 2005. URL <http://linux.sys-con.com/node/49061/>.
- Martin Michlmayr. *Quality Improvement in Volunteer Free and Open Source Software Projects: Exploring the Impact of Release Management*. PhD thesis, University of Cambridge, 2007. URL <http://www.cyrius.com/publications/michlmayr-phd.pdf>.
- Martin Michlmayr. Community management in open source projects. *The European Journal for the Informatics Professional*, X(3):22–26, June 2009.
- Martin Michlmayr, Francis Hunt, and David Probert. Release management in free software projects: Practices and problems. In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Silitti, editors, *Open Source Development, Adoption and Innovation*, pages 295–300. International Federation for Information Processing, Springer, 2007.
- Rich Morin, editor. *Prime Time Freeware for UNIX, Issue 2-2*. Prime Time Freeware, Sunnyvale, CA, July 1993.
- Dare Obasanjo. The myth of open source security revisited v2.0. *developer.com*, March 13 2002. URL www.developer.com/open/article.php/990711/.

- Alexander Osterwalder. *The Business Model Ontology – a proposition in a design science approach*. PhD thesis, l’Ecole des HEC de l’Université de Lausanne, 2004. URL <http://www.hec.unil.ch/aosterwa/PhD/>.
- J. Peek, T. O’Reilly, and M. Loukides, editors. *Unix Power Tools*. O’Reilly, Bantam Books, March 1993.
- Bruce Perens. The open source definition. In Chris DiBona, Sam Ockman, and Mark Stone, editors, *Open Sources: Voices from the Open Source Revolution*. O’Reilly and Associates, Cambridge, Massachusetts, 1999.
- Bruce Perens and Debian. Debian Social Contract. *Web pages*, http://www.debian.org/social_contract, 1997.
- Randy Perry and Al Gillen. Demonstrating business value: Selling to your C-level executives. White paper, IDC, April 2007. Sponsored by Microsoft.
- Rishona Phillips. The bugs are biting. InfoSec reading room, whitepaper, SANS Institute, August 8 2003. URL www.sans.org/reading_room/whitepapers/basics/bugs-biting_1135.
- Jacqueline Rahemipour. Datenverkehrsordnung – OpenOffice.org-Tutorial III: Datenaustausch-Strategien für den Unternehmenseinsatz. *iX*, 5/2010:102–108, may 2010. in German.
- Jacqueline Rahemipour and Florian Effenberger. Office nach Maß – OpenOffice.org-Tutorial II: Firmenrichtlinien umsetzen. *iX*, 5/2010:102–108, may 2010. in German.
- Eric S. Raymond, editor. *The New Hacker’s Dictionary - 3rd Edition*. The MIT Press, 3 edition, October 1996. ISBN 0262680920.
- Eric S. Raymond. The Halloween Documents. web pages, 1998. URL www.catb.org/~esr/halloween/.
- Eric S. Raymond. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O’Reilly Media, first edition, 1999. ISBN 1-56592-724-9.
- Guido Schryen. Is open source security a myth? *Commun. ACM*, 54:130–140, May 2011. ISSN 0001-0782.
- Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. CreateSpace, December 2009. ISBN 1441436200.
- Richard M. Stallman. Who does that server really serve? *Boston Review*, 35(2), March 2010. URL <http://bostonreview.net/BR35.2/stallman.php>. last accessed: August 7, 2011.
- Guy Steele and Eric S. Raymond. The Jargon File. Web pages, <http://www.tuxedo.org/~esr/jargon/>, 2000.

- Vincent Tietz, Gerald Hübsch, and Gregor Blichmann. Cloud-Entwicklungsmethoden: Überblick, Bewertung und Herausforderungen. *Informatik-Spektrum*, 34(4), August 2011. in German.
- Linus Torvalds and David Diamond. *Just for Fun: The Story of an Accidental Revolutionary*. HarperCollins, New York, 2001.
- James Wallace. *Overdrive: Behind Bill Gates' Battle to Keep Microsoft on Top*. John Wiley & Sons, April 1998a. ISBN 0471254134.
- James Wallace. *Overdrive: Bill Gates and the Race to Control Cyberspace*. John Wiley & Sons, May 1998b. ISBN 0471291064.
- James Wallace and Jim Erickson. *Hard Drive: Bill Gates and the Making of the Microsoft Empire*. Harper Paperbacks, May 1993. ISBN 0887306292.
- Michael H. Warfield. Musings on open source security models. *Linuxworld*, April 14 2003. URL www.linuxworld.com/linuxworld/lw-1998-11/lw-11-ramparts.html.
- David A. Wheeler. Why open source software / free software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! article on the Internet, April 2007. URL www.dwheeler.com/oss_fs_why.html.
- Wikipedia. Free and open source software. Web pages, 2010. URL http://en.wikipedia.org/wiki/Free_and_open_source_software.
- Sam Williams. *Free as in Freedom: Richard Stallman's Crusade for Free Software*. CreateSpace, December 2009. ISBN 1441437886.
- David Wong. Secure code. *SecurityFocus.com*, July 2 2002. URL <http://www.symantec.com/connect/articles/secure-coding>.
- Christoph Zott, Raphael Amit, and Lorenzo Massa. The business model: Theoretical roots, recent developments, and future research. *IESE Research Papers*, 3(WP-862), 2010. URL <http://www.iese.edu/research/pdfs/DI-0862-E.pdf>.

5 Quality Assessment of FOSS



by Arne-Kristian Groven, Kirsten Haaland, Ruediger Glott, Anna Tannenberg and Xavier Darbousset-Chong

Each year, large amounts of money are spent on failed software investments. Selecting business critical software is both a difficult and a risky task, with huge negative impact on business if the wrong choices are made. Uncertainty is high and transparency is low, making it hard to select candidate software.

The widespread development and use of Free/Libre and Open Source Software, FOSS, enable new ways of reducing risk by utilising the inherent transparency of FOSS. Transparency is related to the fact that the source code is available on the Internet. In addition, most of the communication about the software takes place on the Internet. Hence, a pool of information is available to anyone wanting to reduce risk when selecting business critical software among FOSS candidates.

Tools and methods for assessing FOSS software, based on measuring data available on the Internet, have been a research issue the last decade. The name FOSS quality (and maturity) model or FOSS quality (and maturity) assessment method appear in the literature to describe such methods. Alternatively, they could also have been referred to as FOSS trust/risk assessment models. There exist two generations of FOSS quality assessment methods, where the first generation was published between 2003 and 2005. About four or five methods were introduced, having a rather limited set of metrics and manual work procedures. In most cases the only software tool support consists of excel-templates for calculations. A second generation of methods was published between 2008 and 2010, following extensive research funding from the European Community. These methods differ from the first generation in increased complexity, both regarding the number of metrics used and the fact that they are semi-automatic approaches with associated software tool support.

In the following text, one first and one second generation FOSS quality model are presented, discussed, and compared with the other. This is done in order to give the reader a brief introduction into such methods; their structure, they work context, their strengths and weaknesses. The intension is not to give a detailed tutorial of any of the methods, but instead to indicate the principles. The text presented here is based on comparative studies and experiments performed in 2009/2010 (Glott et al., 2010; Groven et al., 2010; Haaland et al., 2010).

5.1 Software Quality Models

After briefly introducing traditional software quality models we give a short overview of first and second generation FOSS quality models. The latter will be presented in-depth in the following sections.

5.1.1 Traditional Software Quality Models

Quality is defined in many different ways. It is a rather elusive concept which can be approached from a number of different angles. The various perspectives of quality (Garvin, 1984; Kitchenham and Pfleeger, 1996) include: (i) User view on quality: Focusing on software that meets the users' needs, where reliability, performance/efficiency, maintainability, and usability are core issues. (ii) Manufacturing view on quality: Product quality is derived from conformance to specification and organisations capability of producing software according to defined software processes. Defect-count and staff effort rework costs are examples of relevant issues within this view. (iii) Product view on quality: Focusing on specifying that the characteristics of products are defined by the characteristics (size, complexity, and test coverage) of its sub-parts. Component complexity measures, design, and code measures all fall within this view.

Control and understanding of the quality of software products and their making have been approached the last four decades from roughly two directions; (i) "Quality management approaches" and (ii) "Quality model approaches". Within the category of quality management, we have Deming's quality management approach (Deming, 1988), Crosby's quality management approach (Crosby, 1979), Feigenbaum's approach (Huggins, 1998) which is the predecessor of TQM, and Weinberg's quality management approach (Weinberg, 1994).

Whereas the quality management approaches represent a more flexible and qualitative view on quality, the quality models represent a more fixed and quantitative view (Robson, 2002). At least two directions of quality models exist, where one direction is focusing around either processes or capability level. Following this direction, quality is measured in terms of adherence to the process or capability level. Examples of such quality models are all the variants of the proprietary *Capability Maturity Model* (Paulk et al., 1993), CMM, including CMMI-SE/SW, ISO/IEC 15504 (Loon, 2007), and ISO9001 (International Standards Organisation, 2000). Another direction of quality models is focusing around a set of attributes/metrics used to distinctively assess quality by making quality a quantifiable concept. These include the McCall model (McCall et al., 1977), the Boehm model (Boehm et al., 1976, 1978), and the product quality standard *ISO 9126-1:2001, E* (International Standards Organisation, 2001). ISO 9126 is based on Boehm's and McCall's models. In ISO 9126 six quality characteristics are defined: a) functionality, b) reliability, c) usability, d) efficiency, e) maintainability, and f) portability. Each of these characteristics has a set of sub-characteristics. For example, reliability has the sub-characteristics maturity, fault tolerance, recoverability, and reliability compliance. The measured value of each sub-characteristics gives a metric for the characteristics (Jung et al., 2004).

5.1.2 First Generation FOSS Quality Models

While the traditional software quality models have a history of around four decades, the first FOSS quality and maturity models emerged between 2003 and 2005. While traditional quality models originate in the context of traditional software industry and its proprietary business models, FOSS characteristics are not covered by such models.

Among the first generation FOSS quality models are: (i) the *Open Source Maturity Model*, OSMM Capgemini, provided under a non-free license, (Duijnhouwer and Widdows, 2003); (ii) the *Open Source Maturity Model*, OSMM Navica, provided under the Academic Free License and briefly described by Golden (2004); (iii) the *Qualification and Selection of Open Source software*¹, QSOS, provided by Atos Origin under the GNU Free Documentation License; and (iv) the *Open Business Readiness Rating*², OpenBRR, provided by Carnegie Mellon West Center for Open Source Investigation, made available under the Creative Commons BY-NC-SA 2.5 License. All the above quality models are drawing on traditional models, which have been adapted and extended to be applicable to FOSS.

All models are based on a manual work, supported by evaluation forms or templates. The most sophisticated tool support can be found for QSOS, where the evaluation is supported by either a stand-alone program or a Firefox plug-in, which also enables feeding results back to the QSOS website for others to download. But still, the data gathering and evaluation itself is a manual work process.

As of 2010, none of these FOSS quality models have seen a wide adoption and they can really not be considered a success, despite that the QSOS project shows a slow growth in popularity (Wilson, 2006b). The OSMM Capgemini model has a weak public presence for the open source community; for the OSMM Navica model the web resource are no longer available, while OpenBRR for a long time has had a web site announcing that a new and better version is under way.

The reasons for this lack of success are probably a combination of the following (Groven et al., 2010): (i) The approaches have shortcomings; (ii) the knowledge about the approaches are not properly disseminated; (iii) the success stories are not properly disseminated; and (iv) the business expectations of the originators of these models were possibly unrealistic. But despite of shortcomings and lack of community support, it is our belief that these quality models could play a role when evaluating candidate FOSS. These views are supported in literature, e.g., by Wilson (2006a). There are some success stories, such as the Open University's use of OpenBRR to select a Virtual Learning Environment (Sclater, 2006). The fact that several enterprises³ use OpenBRR, underlines its (potential) role. Further, the simplicity of a first generation FOSS quality and maturity model is intuitively appealing and may have some advantages compared to second generation models.

1. See <http://www.qsos.org/>; accessed November 20, 2010.

2. See <http://www.openbrr.org/>; accessed November 20, 2010. Currently, the original content at this web site is not available, and replaced by a short information page.

3. As an example, the enterprise FreeCode employs OpenBRR in their evaluations of FOSS, see www.freecode.no; accessed November 20, 2010.

5.1.3 Second Generation FOSS Quality Models

Recently, a second generation of FOSS quality models has emerged, partly as a result of several EC funded research projects. They all draw on previous methodologies, both traditional quality models as well as the first generation FOSS quality and maturity models. Two main differences between the first and second generation FOSS quality models are more extensive tool support and more advanced metrics.

Second generation quality models include (i) the *QualOSS quality model*⁴ – a semi-automated methodology for quality model drawing on existing tool support, explained in greater detail in this text; (ii) the *QualiPSo OpenSource Maturity Model (OMM)*⁵, a CMM-like model for FOSS. QualiPSo OMM “focuses on process quality and improvement, and only indirectly on the product quality” (Qualipso, 2009). The project aims at providing supporting tools and assessment process together with the OMM, being a part of a larger EU-initiative which is still under development. QualiPSo draws more strongly on traditional quality models, in this case CMM. Another second generation model is (iii) the *SQO-OSS quality model*⁶ – the Software Quality Observatory for Open Source Software (SQO-OSS) which is a platform with quality assessment plug-ins. SQO-OSS has developed the whole assessment platform from scratch, aiming at an integrated software quality assessment platform. It comprises a core tool with software quality assessment plug-ins and an assortment of user interfaces, including a web user interface and an Eclipse plug-in (Samoladas et al., 2008). The SQO-OSS is being maintained, but the quality model itself is not yet mature, and developers focus mostly on an infrastructure for easy development of plug-ins.

5.2 OpenBRR

The *Open Business Readiness Rating* model, OpenBRR, consists of a set of themes or categories each containing a set of metrics. These categories are spanning the different quality dimensions of an OpenBRR assessment. There are 27 unique and generic metrics to be applied on each types of software to be assessed by the model. In addition, functionality specific metrics have to be tailor-made for each class of software to be assessed.

A high-level view of the usage of the OpenBRR model for evaluating FOSS consists of the following three steps: 1) Creating a shortlist of candidate software to be assessed. 2) Determining the relative importance of the categories and metrics. 3) Manually obtaining the data for the metrics. Step 1 must be performed first, by identifying one or more software candidates, while Steps 2 and 3 may be performed in any order. It is a manual process, and it aims to be complete, simple, adaptable, and consistent.

A spreadsheet template is the only OpenBRR tool support available when creating a business readiness rating.⁷ Measured data are registered in the spreadsheets and BRR scores

4. See www.qualoss.org; accessed November 20, 2010.

5. See www.qualipso.org; accessed November 20, 2010.

6. See www.sqo-oss.eu; accessed November 20, 2010.

7. The template described here is an updated version provided to the authors by Dr. Wasserman from the Center for Open Source Investigation at Carnegie Mellon West, who developed the OpenBRR model in

	<i>Category</i>	<i>Description</i>	<i>Weight</i>
(1)	Functionality	Features offered by the software.	25%
(2)	Operational Software Characteristics	Metrics concerning user experience, security, performance and scalability.	15%
(3)	Service and Support	Metrics describing availability of professional and community support.	25%
(4)	Software Technology Attributes	Metrics describing technical architecture, release cycle and code quality (bug statistics).	10%
(5)	Documentation	Metrics describing the availability and quality of documentation.	10%
(6)	Adoption and Community	Metrics describing the activity of the community and existence of reference installations.	10%
(7)	Development Process	Metrics for stability and quality of project driver and code contributors.	5%

Table 5.1. Categories and weights in the Asterisk BRR

are automatically computed based on these data.

5.2.1 Quality Categories and their Weights

A set of quality categories or dimensions are predefined within OpenBRR. These are shown in Table 5.1. Associated with each category is a set of metrics. All metrics are predefined and generic in their nature, except for the “Functionality” category which requires specific software features of interest to be added into the sheet. As can be read from Table 5.1, weights are also associated with each of the categories. These can be freely set by the BRR evaluator. It is also possible to limit the scope of the evaluation to only cover a subset of the categories.

The category weights should be based on a business case. Table 5.1 illustrates a mission critical usage setting assuming, e.g., that “Service and Support” is very important for business critical applications. Hence, that category was given a high weight.

5.2.2 Metrics, Scores, and Metric Weights

Various metrics are associated with each of the categories in OpenBRR. The number of metrics is relatively small. If we exclude the user-provided metrics associated with the “Functionality” category, altogether 27 unique metrics exist to cover all the remaining six categories. Two metrics are used in two categories and the rest are associated with only one category. Each of these 27 metrics defines: (i) What to measure, and (ii) How to transform the measured values into thresholds or scores. The latter is predefined as test score specifications for each of the 27 metrics. When measuring the various features included in the “Functionality” category, no test score specification exist for each feature.

2005 together with Intel Corporation, Spike Source and O’Reilly Code Zoo. The main change is that the new template has only seven categories, compared to twelve in the first version.

Here the scores have to be set based on subjective evaluations.

To illustrate the metrics we choose one from the sub-category “Security”⁸ which is part of the category “operational Software Characteristics”. The definition of what to measure is described as follows: “Number of security vulnerabilities in the last 6 months that are moderately to extremely critical” having the associated test description “This measures the quality related to security vulnerabilities. How susceptible the software is to security vulnerabilities.” The test score specification is as follows:

1	More than 6 (“Unacceptable”)
2	5 - 6
3	3 - 4
4	1 - 2
5	0 (“Excellent”)

According to this specification five different scores can be given based on the measured data: The best score is given if no security vulnerabilities are found during the last six months with a severity of moderately to extremely critical. If the measurement shows more than six such vulnerabilities, the worst score (1) is given. The five-value scoring range is typical among the 27 generic metrics. But sometimes a three-value range is used, as illustrated in the following metrics: “Difficulty to enter the core developer team”, with the following test description: “To ensure software quality, mature projects must be selective in accepting committers. New projects often have no choice”. Here the test score specification is defined as follows:

1	Anyone can enter;
3	Rather difficult, must contribute accepted patches for some time;
5	Only after being active outside-committer for a while.

While there are no possibilities within OpenBRR for the evaluators to change any of the 27 test score specifications, one can freely set the relative importance for each set of metrics within any of the categories.

5.2.3 OpenBRR Work- and Information Flow

Before starting an OpenBRR evaluation, it has to be configured to fit the business context in which the evaluation shall take place. This is done by setting weights, both between the categories and between the metrics within each category. In addition, the feature set of interest has to be identified. Profound knowledge on requirements and technology is needed here. Each feature of interest is registered in the “Functionally” category of the spreadsheet.

Exactly what type of data to look for is defined by each of the 27 generic metrics. The generic metrics are predefined within the method, to be used in the evaluation of all types of FOSS. When a measurement is registered in the OpenBRR sheet, a score will be calculated according to the predefined thresholds (test score specifications). The relative importance of the metrics within a category is up to the evaluator to decide, by setting

8. Some of the categories are divided into sub-categories. See Table 5.2 for more details

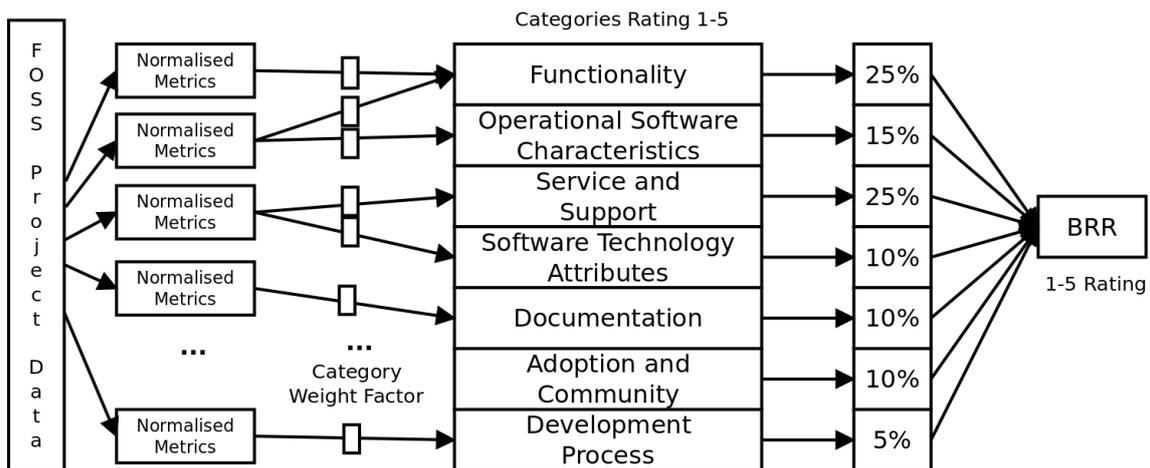


Figure 5.1. The Business Readiness Rating Model

weights on each of the metrics. Likewise, the relative importance between each category can be decided by setting category weights. Based on the measured data, metrics scores are calculated and aggregated for each category. Each item under “Functionality” is subjectively evaluated without any associated test score specification. This part of the assessment is based on the technical knowledge of the evaluator. Each feature receives a score within a certain range that is further aggregated into an overall functionality score. An overall score is finally aggregated from the category scores and their relative importance as shown in Figure 5.1.

Since the OpenBRR model does not provide any tools for data mining, all data must be collected manually. An OpenBRR assessment is fully depending on the knowledge and ability of the evaluator to find the right data on the Internet. Various mailing lists, bug trackers, databases, and web sites for harvesting data must be identified, both specifically related to the FOSS project at hand or third party. Examples of the latter are, e.g., www.secunia.com as a source for information on number of critical security issues or Amazon.com as a source for information about publications.

Quality assurance should always be performed on a BRR before it is considered completed. Errors in the formulas of the spreadsheet can easily be introduced, weights miscalculated, information sources excluded, etc.

5.3 QualOSS

QualOSS provides a high-level methodology for benchmarking the quality of FOSS. Main quality focus for the benchmarking are the “Evolvability” and “Robustness” of FOSS (De-prez et al., 2008): 1) “Robustness” is defined to be the capability that the FOSS endeavour displays in solving past and current problems. 2) “Evolvability” is defined to be the capability that the FOSS endeavour will likely display in solving future problems.

QualOSS uses the term “FOSS endeavour” instead of FOSS project. A FOSS endeavour is defined by the following four elements: 1) A set of work products, 2) the FOSS community creating, updating and using these work products, 3) the tools used to act on

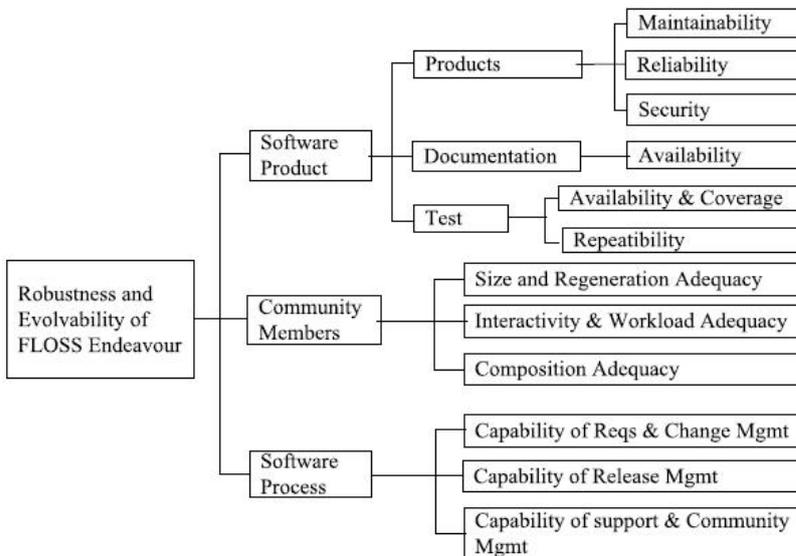


Figure 5.2. Structure of the QualOSS Standard Assessment

these work products or to build or run the software product, and 4) the set of development processes executed by the community, these processes include rules and a division of labour accepted and followed by community members when interacting and creating work products” (Ruiz and Glott, 2009). The third element has so far not materialised into QualOSS.

Figure 5.2 illustrates the structure of the *QualOSS Standard Assessment Method*, starting with the defined quality focus (robustness and evolvability) as the root node, further decomposed into the FOSS endeavour elements (software product, community members, software process), and ending up with a set of subgoals as the leaves. Various metrics are further associated with each of the leaf characteristics.

The overall purpose is to evaluate the degree of risk for selected (or all) leaf characteristics, related to a selected context with specified viewpoints. The QualOSS Standard Assessment Method represent one specific predefined configuration, context (*Usage*=integration in a product, *Mode*=product comparison, *Collaboration*=full FOSS collaboration) with a set of viewpoints (long term management viewpoint, short term management viewpoint, long term technical viewpoint, short term technical viewpoint). The intension behind the possibility of making configurations is to “tune” the measurement to specific business cases. Depending on the configuration, only the most relevant metrics will be used. But since only one standardised configuration exist, the *QualOSS Standard Assessment Method*, more details regarding configurations will not be pursued in any more depth here.

Goal Question Metrics

QualOSS uses the GQM, Goal Question Metrics, invented by Basili (1992). It associates a GQM template with each of the leaf characteristics in Figure 5.2. These are “Maintainability”, “Security”, “Reliability”, “Availability”, “Availability and Coverage”, “Repeatability”.

ity”, “Size and Regeneration Adequacy”, “Interactivity and Workload Adequacy”, “Composition Adequacy”, “Capability of Requirements and change management”, “Capability of Release Management”, and “Capability of Support and Community Management”.

The leaf characteristics represent assessment (sub-) goals and, based on the configuration, a set of questions is associated with each (sub-)goal. For an assessment goal on “Maintainability” from a product manager’s viewpoint, the following questions are defined in QualOSS: *a)* What is the percentages of enhancements proposal that get accepted? *b)* What is the rapidity with which accepted enhancements are implemented? *c)* What is the percentage of changes in the code between major releases? *d)* What is the percentage of changes to public interfaces in the code (external API) between major releases? *e)* What is the evolution in code volumetry between various releases of the code over time (in chronological order)?

Associated with each question is a set of (one or more) risk indicators. The question “What is the percentages of enhancements proposal that get accepted?” is associated with the following risk indicator: *i)* *green* colour indicates that 10% of the enhancement proposals are accepted; *ii)* *yellow* colour indicates that between 5% and 10% of the enhancement proposals are accepted; *iii)* *red* colour indicates that between 2% and 5% of the enhancement proposals are accepted; and *iv)* *black* colour indicates that less than 2% of the enhancement proposals are accepted.

Various metrics are associated with each of the risk indicators. In the example above the following two metrics are defined: 1) number of enhancement proposals; and 2) number of accepted enhancement proposals. Additionally, both an artifact type, a data source type, and a specification of a measurement procedure are defined for each of the metrics. Risk indicators are equivalent to the OpenBRR test score specifications, predefined by the method.

5.3.1 QualOSS Work- and Information Flow

The first step in a QualOSS assessment is to configure the assessment according to viewpoints, contexts, etc.⁹. This results in a predefined (sub-)set of questions, associated risk indicators, and metrics. For each leaf characteristic in Figure 5.2 a GQM template must be filled in during the measurements. Like in OpenBRR the template is a spreadsheet.

Measurements are performed according to the defined metrics. These are highly automated using a number of supporting tools, such as CVSanaly¹⁰ and Bitcho¹¹. However, a good portion of the measurements, especially on documentation, has to be done manually. The measurement results are documented in spreadsheets that are filled in automatically or manually. The resulting scores from measurements related to each of the leaf characteristics are finally aggregated to form an overall risk indication. Assessment results in QualOSS are finally presented in a graph, as illustrated in Table 5.3.

9. QualOSS Standard Assessment 1.1 is a predefined configuration.

10. See <http://cvsanaly.tigris.org/>; accessed November 23, 2010.

11. See <http://tools.libresoft.es/bicho>; accessed November 23, 2010.

5.4 Experiences and Results from Assessments

In 2009 we performed assessments using both OpenBRR and QualOSS on the PBX (Public Branch Exchange) Voice over IP software named *Asterisk*¹². In the following, results and experiences are presented.

5.4.1 OpenBRR Assessment

Two different OpenBRR assessments for the same target software *Asterisk* were performed during 2009. The first assessment¹³ formed the baseline. It was performed by (two) FOSS professionals with a high level of knowledge about OpenBRR, here denoted as Evaluator *B*. This resulted in a total rating of 4.24 (out of 5). The second assessment was a partial assessment. Hence, no overall score was given. This evaluation was performed, as a student project in a master level course¹⁴ at the University of Oslo¹⁵, here denoted as Evaluator *E*.

The results of the two assessments showed quite similar results. In Table 5.2, all scores made by both assessments are listed under "Evaluation results". The scores for each metric range from 1 to 5, where 1 is "Unacceptable" and 5 is "Excellent". The metrics are grouped per sub-category, respectively per category if no sub-category is available, as defined in the OpenBRR template. The columns "E." and "B." denote the results from the respective evaluator.

Looking at the scores, we find that 17 out of 29 metrics (including two metrics used twice) resulted in the same rating. For four metrics the difference were off by 1 in the rating. Two of the metrics were not completed by Evaluator *E*, while the last six metrics were off by 2. Note that some of the metrics only have three choices with scores 1, 3, and 5. Hence, some of these measurements do not differ as much anyway.

We observed that Evaluator *E* did not feel confident enough to put scores on two of the metrics. Also the functionality part of the OpenBRR was rather difficult to assess for him. As a consequence, the evaluation by Evaluator *E* did not result in an overall BRR ranking. Evaluator *B* ended up with a score of 3 on the functionality part, which means *acceptable*.

The weights set by Evaluator *B* were reused by Evaluator *E* without any modifications. The non-functional categories had a weight of 75% of the total evaluation while the functionality part was set to a 25% weight of the total evaluation.

During an OpenBRR assessment the key success criteria is the evaluators' ability to find trustworthy and complete sources of information on the Internet. In addition, proper filtering of information might be a challenge. Evaluator *E* experienced some difficulties

12. See <http://www.digium.com/en/>.

13. As part of the EUX2010sec research project, partly funded by The Research Council of Norway (project number 180054).

14. INF5780, autumn 2009.

15. By then the current software version had changed from 1.4.25 to 1.4.26, causing some discrepancy in some sub-categories.

<i>No</i>	<i>Category</i>	<i>Metrics</i>	<i>E.</i>	<i>B.</i>
1	2.1 Usability	End user UI experience	1	3
2		Setup time for pre-requisites	4	5
3		Time for vanilla installation and config.	4	4
4	2.2 Security	# (moderate to extremely critical) security vulnerabilities, past 6 months	3	4
5		# security vulnerabilities unpatched	-	5
6		Dedicated security info available	5	5
7	2.3 Performance	Performance Testing and Benchmark Reports available	5	3
8		Performance Tuning & Configuration	5	5
9	2.4 Scalability	Reference deployment	5	5
10		Designed for scalability	3	5
11	3 Service and support	Avg. volume, general mailing list, past 6 months	5	5
12		Quality of professional support	5	5
13	4.1 Architecture	Are 3rd party Plug-ins available?		5
14		Public API / External Service	5	5
15		Enable/disable features by config.	3	5
16	4.2 Quality	# minor releases, past 12 months	1	1
17		# point/patch releases, past 12 months	1	3
18		# open bugs, past 6 months	4	5
19		# bugs fixed, past 6 months (compared to # bugs opened)	5	5
20		# P1/critical bugs opened	1	2
21		Average bug age for P1 in last 6 months	1	1
22	5 Documentation	Existence of various documents.	5	5
23		User contribution framework	5	5
24	6.1 Adoption	# of books at amazon.com	5	5
25		Reference deployment	5	5
26	6.2 Community	Avg. volume general mailing list, past 6 months	5	5
27		# unique code contributors, past 6 months	4	4
28	7 Development process	Project Driver	4	4
29		Difficulties to enter core developer team	3	5

Table 5.2. Comparison of Asterisk BRR results

included, e.g., in the quality-part, on how to sort (high volume) bug archives to find those who are solved and closed and those which are still open.

Some of the assessment results are worth a comment. There are, e.g., low scores on some of the quality metrics. The “average bug age for P1 bugs the last 6 months” got the score 1 in both evaluations. Since Asterisk is a complex, business critical software, the identification and correction of errors are probably more time consuming than for less complex, non-critical software. Faced with the test score specification for this metric, such factors are not considered. Since the overall quality of Asterisk is perceived by the users to be good, it seems that complex systems like Asterisk on a few metrics are punished by OpenBRR compared to less complex systems.

Asterisk and its community score high on, e.g., the metric “Difficulty to enter the core development team”, resulting in the score 5 since this is possible “Only after being active outside committer for a while”. Another metric called “Project Driver”, rates Asterisk high (score 4) since it has a “Corporation” rather than “Groups” or “Individuals”. Only “Independent foundations supported by corporations” gives a higher score, 5. Some might disagree on this rating scale. Less controversial, Asterisk scores high on community, adoption, and documentation metrics since it is a widespread piece of software with a high activity level.

Another experience from using OpenBRR suggests that it is very easy to manipulate the overall result by changing weights in one way or another. An OpenBRR has to be tuned towards customers and their requirements and it has to be kept the same for all comparable software candidates.

The main conclusion, after comparing two OpenBRR assessments on the same software, is the following: Despite Evaluator *E*'s lack of experience in performing OpenBRR evaluations, the results from the Asterisk Business Readiness Rating indicate a quite high degree of consistency between the two evaluation. Assuming less experience from Evaluator *E* compared to Evaluator *B*, he was able to perform quite many ratings which for some are similar to the benchmark results made by Evaluator *B*.

5.4.2 QualOSS Assessment

QualOSS and OpenBRR both cover different views of quality, (i) the product view on quality, (ii) the manufacturing, or process, view on quality, and also to some smaller extent (iii) the user view on quality. But the differences in the two approaches are obvious: While OpenBRR is performed manually, having only a spreadsheet for registration of results and calculation of scores, the QualOSS model relies on automation using software tool support to capture data on the Internet. But, QualOSS also relies on manual processes whenever suitable tools are unavailable. This was the case for some of the measurements when assessing Asterisk¹⁶. There is also a difference in the output of the two quality as-

16. As part of both the EUX2010sec research project, partly funded by The Research Council of Norway (project number 180054) and the QualOSS research project, partly funded by the European Commission, Information Society Technologies, (project number 033547).

Documentation & Evolvability of Endeavour AVG 1.873	Work product AVG 1.32	Product AVG 2.216	Maintainability AVG 1.596
		Documentation AVG 1.333	Reliability AVG 2.1
			Security AVG 2.682
	Community mbrs AVG 2.282	Test AVG 0.5	Availability AVG 1.333
			Test Availability and Coverage AVG 0.5
		Test Repeatability AVG 0.5	
	Software processes AVG 2.017	Size and Regeneration Adequacy AVG 3	
		Interactivity and Workload Adequacy AVG 1.563	
Capability of requirements & change mngt. AVG 2.333			
		Capability of release management AVG 1.7	

Legend:^a

High risk [0, 1[Medium risk [1, 2[Small risk [2, 3[Negligible risk [3, 4[
---------------------	-----------------------	----------------------	---------------------------

a. Note that we use shades of gray to visualise the risk rather than using the colours green, yellow, red, and black, as specified by QualOSS.

Table 5.3. Risk assessment tree for Asterisk 1.4.26

assessment models: while OpenBRR gives a *score*, QualOSS indicates *trends*.

As illustrated in Table 5.3, the composite result of the QualOSS quality and risk assessment denotes Asterisk version 1.4.26 as a medium risk for businesses. Figure 5.2 shows the QualOSS structure with an aggregation of risk values, from right to left.

Both assessment approaches reacted on the high number of minor releases and patches in the Asterisk 1.4.x product line. Quite a high number of these minor releases and patches are produced to solve security issues based on reported vulnerabilities. This makes it, in general, more difficult to maintain a running Asterisk system from the perspective of a user organisation and its system administrator. Both QualOSS and OpenBRR produce negative scores here as one could expect, but from the perspective of an Asterisk system administrator, the practical implications might not be that dramatic or time consuming: Apart from the core call processing functionality, which is establishing, maintaining, and ending connections, there are many options that can either be turned on or off at an Asterisk application. Therefore, each vulnerability alert has to be validated against the functionality of the running system to identify the need for maintenance.

Regarding documentation, OpenBRR gave a good score while QualOSS gave credit for documentation, but asked for more detailed design and system documentation to be satisfied. Taking a closer look at this finding, it is not possible to, e.g., find a diagrammatic presentation of the core functionality of Asterisk. No design documentation is found either, at least not for Asterisk 1.4.x which was the version assessed by us. In the case of

Asterisk 1.6, online reference documentation for Asterisk version 1.6.1.6 is available¹⁷, but the quality of this has not been analysed any further here.

The most critical output of the QualOSS assessment was the lack of a holistic and structured test regime. This seems to be correct at the time of the assessment in November 2009. However, there is also a possibility that some of the test results have either not been found or have not been made public. The latter may be true for a set of interoperability tests between Asterisk and, e.g., other SIP¹⁸ based devices¹⁹. From each SIPit, Session Initiation Protocol Interoperability Test²⁰, there are non-public information regarding interoperability test results. Regarding performance testing, some information is available, e.g., from third parties. But the information is not extensive. About two months after the QualOSS evaluation was performed Digium announced²¹ increased focus on an Asterisk test framework consisting of the following components: *a*) peer reviews; *b*) unit testing through a new API in Asterisk trunk for writing unit tests within the code; *c*) an external test suite is about to be created; and *d*) regression testing in combination with continuous code integration using Bamboo. This is a clear indication that QualOSS did identify something that was really missing at the time of the assessment.

5.5 Comparing the Assessment Methods

QualOSS and OpenBRR both cover different views of quality, (i) the product view on quality, (ii) the manufacturing, or process view on quality, and, to some smaller extent, (iii) the user view on quality.

When the scope is defined, QualOSS has a large set of predefined metrics and indicators based on GQM, the Goal Question Metrics approach. OpenBRR has a much smaller metrics set, containing 27 different metrics, which are predefined like for QualOSS. However, flexibility arises in OpenBRR when defining the feature set for the Functionality category, both in choosing the actual features (whether to include them as standard or extra), and setting their importance (1-3). This involves human experts into the OpenBRR process. Such type of interaction is not present in the QualOSS assessment, where detailed metrics (e.g., involving coding standards) are defined (at least for some programming languages).

While the QualOSS assessment is a highly automated measurement and uses a number of measurement tools, OpenBRR is based solely on the skills of the evaluators. There is also a difference in the output of the two quality assessment models: while OpenBRR outputs a score, QualOSS also outputs trend indications, e.g., the evolution of the number of lines of code between releases.

The risk of basing the whole assessment on manual work is that critical information can

17. See <http://www.asterisk.org/docs>; accessed November 23, 2010.

18. SIP or Session Initiation Protocol is the de facto signalling standard in VoIP communication.

19. according to a person close to the core development team.

20. SIPit is a week-long event where various SIP implementations are assembled to ensure they work together.

21. See <http://lists.digium.com/pipermail/asterisk-dev/2010-February/042387.html>; accessed August 24, 2011.

be missed. This is also the case for QualOSS, especially in the cases where no suitable tools are present. Then the options are either to perform the assessment on a manual basis or to do the assessment without full coverage of topics. Since the metrics and measurements are more complex than for OpenBRR the last option might sometimes be the right one. Whenever the tool support is working as intended the QualOSS is a source of more insight compared to a method like OpenBRR.

The role of proper quality assurance should be emphasised for both models, including interviews and discussions before and after the assessment. This in order ensure that the assessment methodology captured the relevant items, and to check if the results of the highly automated QualOSS assessment are good and understandable enough to convince people with expertise knowledge of the FOSS endeavours under scrutiny.

In the case of OpenBRR, it is assumed by the model that there is a real need and specific business case as basis for the rating to answer questions like: Who is the customer? What are his/her needs? What is the level of technical knowledge of the customer? What is the available or preferred technical platform to run the software on? Without the answers to these questions, the final score becomes too much a product of the opinions and assumptions of the evaluators, especially obvious when choosing functionality set, evaluating the user experience, and of course setting all the weights for relative importance. The QualOSS Standard Assessment (version 1.1.), which was used in our case, did choose to configure the evaluation towards the needs of a company making business of Asterisk services to end user organisations. But context granularity and fine tuning prior to the assessment could also be higher in this case.

Another challenge and potential problem when working with measurements and metrics is to define the difference between a good result, a bad result, and a neutral result. In the case of metrics related to release cycles in “Software Technology Attributes: Quality” in OpenBRR, they might be too rigid in the view of preferable release cycles. The same applies to QualOSS, when it comes to, e.g., reporting of bugs and vulnerabilities. A trend indicating a rise in bug or vulnerability reporting has several potential interpretations, and all of them are not necessarily negative. Asterisk has experienced extreme growth in number of users the last couple of years. As a consequence, more functionality options have been explored and more hidden errors are found. A challenge for assessment models like QualOSS and OpenBRR is not to punish more complex systems and systems with a large user community. Large projects with active communities will probably get many bug and vulnerability reports while a small project with very few users may not get many. This does not in any way mean that the smaller project is more business-ready or mature. The assessment results on bug and vulnerability reporting should be calibrated against the size of the user community, not only the developer community. A rising trend in reporting might indicate a rise in users, which is not necessarily bad.

The question whether or not the second generation quality model can outperform the first generation model can only be answered with ambiguity. Both quality models have different strengths and weaknesses.

5.6 Concluding Remarks

We have presented two so-called FOSS quality and maturity models aimed at assessing quality and risks related to FOSS. Results from practical application of the two methods have also been presented and discussed, using OpenBRR and QualOSS. Both models (methods) are quantitative, based on data measurements related to predefined metrics. The data sources, covering both the software and its community, are reachable on the Internet. Based on the actual measurements (data collection on the Internet), scores are computed according to predefined score schemes. The aim of both models is to assess the quality and the risks associated with some piece of FOSS software, intended to be used in a specified business context.

OpenBRR allows assessment of a limited set of quality metrics, based on manual data collection. QualOSS, in contrast, involves hundreds of quality metrics. Here, supporting software tools play a prominent role in the data collection. Both models can to some extent be configured towards certain business needs: For OpenBRR by altering the weights of quality characteristics and their associated metrics, and by the addition of a feature list. For QualOSS by configuring the GQM template for each of the leaf characteristics according to predefined viewpoints, modes, and usage value sets.

Based on our experiments we find OpenBRR to be a useful tool with small resource requirements and low time consumption. It needs general knowledge about where to find information on the Internet combined with deep domain knowledge on the part covering functionality. The QualOSS assessment is a highly automated measurement and uses several software measurement tools. Expert skills on functionality are not needed here, compared to OpenBRR. But there has to be experts on the collection tools present. Whenever the tool support is working as intended, the QualOSS is a source of more insight compared to a method like OpenBRR. Whenever the automated tool support is not sufficient, which happened in parts of the QualOSS assessment, we needed to perform relatively time-consuming manual work.

Overall, it appears that human expertise, especially knowledge of context conditions and development trends with a FOSS endeavour, is decisive for the usability of both quality models. OpenBRR relies on this input by design. QualOSS tried to largely eliminate such direct input on the measurement process but, occasionally, seems to rely on it when tools are not available or when the results of the assessment must be interpreted.

Unfortunately, the reported OpenBRR activities are low and the community inactive. This is disappointing as it seems to be potentially a useful tool with small resource requirements. Similarly, the community support for QualOSS has still not reached its full potential, and there is scope to further develop this methodology. Time will show if an active community will grow around QualOSS or be regenerated around OpenBRR, or if another quality model will appear. It is at in any case clear that there is a real need for sound quality models in the market, helping actors make their decisions.

Acknowledgment

The authors would like thank Dr. Wasserman from the Center for Open Source Investigation at Carnegie Mellon West for providing the updated OpenBRR spreadsheet.

References

- Victor R. Basili. Software modeling and measurement: the goal/question/metric paradigm. Technical Report UMIACS TR-92-96, College Park, MD, USA, 1992.
- B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering, ICSE '76*, pages 592–605, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press. URL <http://portal.acm.org/citation.cfm?id=800253.807736>.
- Barry W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merritt. *Characteristics of Software Quality*. North-Holland, Amsterdam, 1978.
- Philip B. Crosby. *Quality is free : the art of making quality certain*. McGraw-Hill, New York, 1979.
- W. Edwards Deming. *Out of the crisis: quality, productivity and competitive position*. Cambridge University Press, 1988.
- Jean-Christophe Deprez, Kirsten Haaland, and Flora Kamseu. QualOSS methodology & QualOSS assessment methods. Deliverable D4.1 to the European Commission, QualOSS project, 2008. URL http://www.qualoss.org/about/Progress/deliverables/WP4_Deliverable4.1_submitted.pdf.
- Frans-Willem Duijnhouwer and Chris Widdows. Open source maturity model. *Capgemini Expert Letter*, 2003. URL http://pascal.case.unibz.it/retrieve/1097/GB_Expert_Letter_Open_Source_Maturity_Model_1.5.31.pdf.
- David Garvin. What does product quality really mean? *Sloan Management Review*, 26: 25–45, 1984.
- Ruediger Glott, Arne-Kristian Groven, Kirsten Haaland, and Anna Tannenber. Quality models for free/libre open source software — towards the silver bullet? *Software Engineering and Advanced Applications, Euromicro Conference*, 0:439–446, 2010.
- Bernard Golden. *Succeeding with Open Source*. Addison-Wesley Professional, 1st edition, August 2004. ISBN 0321268539.
- Arne-Kristian Groven, Kirsten Haaland, Ruediger Glott, and Anna Tannenber. Security measurements within the framework of quality assessment models for free/libre open source software. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA '10*, pages 229–235, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0179-4.

- Kirsten Haaland, Arne-Kristian Groven, Kirsten Haaland, Ruediger Glott, and Anna Tanenberg. Free/Libre Open Source Quality Models — a comparison between two approaches. *4th FLOS International Workshop on Free/Libre/Open Source Software*, July 1-2 2010.
- Lawrence P. Huggins. Total quality management and the contributions of A.V. Feigenbaum. *Journal of Management History*, 4(1):60–67, 1998.
- International Standards Organisation. Quality management systems – requirements. Technical Report ISO/IEC 9001:2000, 2000.
- International Standards Organisation. Software engineering – product quality, part 1: Quality model. Technical Report ISO/IEC 9126-1:2001, 2001.
- Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: A survey of iso/iec 9126. *IEEE Software*, 21(5):88–92, 2004. ISSN 0740-7459.
- Barbara Kitchenham and Shari Lawrence Pfleeger. Software quality: The elusive target. *IEEE Softw.*, 13:12–21, January 1996. ISSN 0740-7459. doi: 10.1109/52.476281.
- Han van Loon. *Process Assessment and ISO/IEC 15504: A Reference Book*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 0387300481.
- Jim A. McCall, Paul K. Richards, and Gene F. Walters. Factors in Software Quality. Volume I, II, and III. Technical Report NTIS AD-A049-014, AD-A049-015, AD-A049-055, Nat'l Tech.Information Service, November 1977.
- Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model, version 1.1. *IEEE Software*, 10:18–27, 1993. ISSN 0740-7459.
- Etiel Petrinja, Alberto Sillitti, and Giancarlo Succi. Comparing openbrr, qsos, and omm assessment models. In Pär Ägerfalk, Cornelia Boldyreff, Jesús González-Barahona, Gregory Madey, and John Noll, editors, *Open Source Software: New Horizons*, volume 319 of *IFIP Advances in Information and Communication Technology*, pages 224–238. Springer Boston, 2010.
- Qualipso. CMM-like model for OSS. Course material, 2009. URL <http://www.qualipso.org/node/175>.
- Colin Robson. *Real world research: a resource for social scientists and practitioner-researchers*. Blackwell Publisher Ltd., 2002.
- José Ruiz and Rüdiger Glott. Result of case studies. Deliverable D5.3 to the European Commission, QualOSS project, 2009. URL <http://www.qualoss.org/deliverables/qualoss%20test2.rtf>.
- Ioannis Samoladas, Georgios Gousios, Diomidis Spinellis, and Ioannis Stamelos. The SQO-OSS quality model: Measurement based open source software evaluation. In Ernesto Damiani and Giancarlo Succi, editors, *Open Source Development, Communities*

and Quality — *OSS 2008: 4th International Conference on Open Source Systems*, pages 237–248, Boston, September 2008. IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, Springer. doi: 10.1007/978-0-387-09684-1-19. URL <http://www.dmst.aueb.gr/dds/pubs/conf/2008-OSS-qmodel/html/SGSS08.htm>.

Niall Sclater. Enhancing and embedding a mission-critical open source virtual learning environment. In *Open Source and Sustainability*, Said Business School, Oxford, 10-12 April 2006. URL <http://www.oss-watch.ac.uk/events/2006-04-10-12/presentations/niallsclater.pdf>.

Gerald M. Weinberg. *Quality software management (vol. 3): congruent action*. Dorset House Publishing Co., Inc., New York, NY, USA, 1994. ISBN 0-932633-28-5.

James A J Wilson. Business readiness rating. Web pages, 2006a. URL <http://www.oss-watch.ac.uk/resources/brr.xml>.

James A J Wilson. Open source maturity model. Web pages, 2006b. URL <http://www.oss-watch.ac.uk/resources/osmm.xml>.

6 Open Licensing



by Wolfgang Leister

In this chapter we introduce and discuss the large area of licensing of different kinds of content and goods, i.e., everything that can be put under a copyright. As discussed earlier, CBPP covers phenomena such as software, content, hardware, designs, databases, public sector information, and scientific data. For many areas, the ideas of universal access, modification, and distribution of content, designs, and databases is desired. The technological progress, e.g., the introduction of *apps*¹, make it a necessity to share data.

Content should be put under a proper license, also when giving open and free access. Just providing content without license information is legally allowed, but will create confusion to how, and on what terms this content can be used. This is, e.g., relevant for public sector data. When content is properly licensed, it can be used properly by all parties. Using free and open licenses gives the possibility for everybody to build upon the knowledge of others, thus boosting innovation.

Licenses for free and open source software (FOSS) have been already discussed in Section 4.3. We discussed the term of *openness* leading to licenses such as the GPL or the BSD licenses which build on copyright, as well as *public domain*, where the copyright is waived by the creator. Software licenses are designed to be applied to software, taking into account terms like *executable code*, *source code*, or *software libraries*. Software licenses are likely not very suited for other types of content, since some of the software licenses' terms do not necessarily make sense for non-software.

Licensing is built on copyright legislation which gives the creator a time-limited right to decide how to use the creation. After the copyright has passed, the content will be in the public domain. When the creator releases content under a license, he or she still retains the copyright. Open licenses that are *non-exclusive*, therefore, make it possible that the same content can be licensed at the same time under different conditions by the creator. The open licenses do not oppose copyright; in contrary, they build upon it.

It is important to distinguish between *access* to content, such as viewing, and *use* of content, such as distributing, altering, and distributing altered content. The copyright laws address the use, but not pure access. While the copyright laws cannot be a basis for access control, a copyright holder can decide whether to distribute content with a closed or open license. The use of content includes *a*) distributing the content unaltered;

1. With *apps* we denote small applications that are downloadable to run on smartphones, tablets or in web browsers. These apps often offer the user a functionality to retrieve content, make some processing, and present the result on the screen.

With the advent of the digital revolution and the Internet, it is suddenly possible to distribute works in a variety of formats of a high, often professional quality; to work collaboratively across contexts; and to create new, derivative or collective works – on a global level, in a decentralised manner, and at comparatively low cost. This presents an opportunity for an enormous and unprecedented stimulation of creativity and production of knowledge. As more and more people are interconnected and communicating, it becomes easier to obtain exactly the content one needs or want and to complete tasks and solve problems by the cooperation this interconnection enables. The convergence of technologies and media also create multiple new possibilities for creating derivatives of existing works – for example, remixes and mashups.

The downside of these exciting new developments and possibilities is that the new technologies can also be used to violate the rights of copyright owners as they are currently defined. In turn, major right holders have reacted to this by a fourfold strategy: (1) by trying to prevent the deployment of technologies that can be put to infringing uses; (2) by developing tools that enable them to manage their rights with an amount of precision hitherto unknown and unthinkable: digital rights management and technological protection measures against unauthorised copying; (3) by successfully lobbying for support of these technological measures through legal restrictions; and, (4) by starting huge publicity campaigns designed to teach young people that they must keep their hands off copyrighted material.

Source: <http://wiki.creativecommons.org/FAQ>; accessed August 14, 2011; © Creative Commons, CC BY.

Frame 6.1. Problem description from the CC FAQ

b) distributing the content in a *collection* together with other works, where the content as such is unaltered albeit editorial changes; c) distributing adaptations and *derivative work* from the original work; and d) performing and distributing *produced work*, using content from the original work.

Copyfraud is a term used by Mazzone (2006) to describe the use of false claims of copyright to attempt to control works not under one's legal control. He describes copyfraud to include 1) claiming copyright ownership of public domain material; 2) imposition by a copyright owner of restrictions beyond what the law allows; 3) claiming copyright ownership on the basis of ownership of copies or archives; 4) and claiming copyright ownership by publishing a public domain work in a different medium. Mazzone argues that copyfraud is usually successful because there are few and weak laws criminalising false statements about copyrights and lax enforcement of such laws.

There are several initiatives, organisations, and large-scale projects looking into specific licensing problems that relate to the *digital public domain*²; *open access policies*; exceptions and limitations to copyright such as fair use and fair dealing in common law systems, or *orphan works*³.

6.1 Creative Commons

The *Creative Commons*⁴ (CC) provide creators and licensors with a simple way to say what freedoms they want their creative work to carry, as outlined in Frame 6.1. This,

2. See <http://www.communia-project.eu/about>; accessed August 24, 2011.

3. See Frame 6.2 in Section 6.3.

4. See creativecommons.org; accessed August 11, 2011.

in turn, makes it easy to share, build upon creative work, and to reserve some rights while releasing others, based on copyright legislation. Formally, *Creative Commons* is a charitable corporation in the US. James Boyle, Michael Carroll, Lawrence Lessig, Hal Abelson, Eric Saltzman, and Eric Eldred founded the CC in 2001.

The copyright laws create the traditional “all rights reserved” setting for content, that is creating an environment where a licensor (rights holder) grants rights for an identified resource (asset) to a principal (party) under certain conditions.⁵ This license is valid as long as the copyright can be applied. After that, the content will go into the public domain. While traditional licenses based on copyright are tailored to commercially exploited content, the CC licenses define a standardised way of granting copyright permissions to creators’ work for commons. The CC license and tools rooted in copyright law are suited for the growing digital commons content that is made to be copied, distributed, edited, remixed, and built upon.

Be aware that all of the CC licenses contain a disclaimer of warranties, so there is no assurance whatsoever that the licensor has all the necessary rights to permit reuse of the licensed work. This disclaimer means that the licensor is not guaranteeing anything about the work, including that she or he owns the copyright to it, or that she has cleared any uses of third-party content that her work may be based on or incorporate.

Please note that parts of this text are adapted from from the Creative Commons web site, which is licensed under a Creative Commons Attribution 3.0 License (CC-BY). As we will see from the following discussion, this license makes it legally possible for the authors of the current text to cite from the Creative Commons web site, and mix it with our own content, without asking for the copyright holder’s permission first, as long as credit to the originator of the site is given.

6.1.1 Use of the Creative Commons License

When using a Creative Commons license, the creator, called *licensor* in the Creative Commons-terms, retains copyright while allowing others to copy, distribute, and make some uses of the licensed work under the conditions that the licensor finds appropriate. Creative Commons licenses work around the world, and last as long as the applicable copyright lasts. While the licensor can decide to license the content also under other terms additionally, a once granted Creative Commons license is non-revocable.

The Creative Commons licenses do not affect freedoms that the law grants to users of creative works otherwise protected by copyright, such as exceptions and limitations to copyright law like fair use⁶. Licensees must credit the licensor, keep copyright notices intact on all copies of the work, and link to the license from copies of the work. Licensees

5. See www.contentguard.com/drmwhitepapers/CGWP-FinalEng.pdf; accessed August 11, 2011.

6. *Fair use* is a limitation and exception to the exclusive right granted by copyright law to the author of a creative work. Fair use provides for the legal, unlicensed citation or incorporation of copyrighted material in another author’s work under certain conditions. Examples of fair use include commentary, criticism, news reporting, research, teaching, library archiving and scholarship. See en.wikipedia.org/wiki/Fair_use; accessed March 5, 2012.

Terms of a Creative Commons license:

Attribution. You let people copy, distribute, display, perform, and remix your copyrighted work, as long as they give you credit the way you request. All CC licenses contain this property.

NonCommercial. You let people copy, distribute, display, perform, and remix your work for non-commercial purposes only. If they want to use your work for commercial purposes, they must contact you for permission.

ShareAlike. You let people create remixes and derivative works based on your creative work, as long as they only distribute them under the same Creative Commons license that your original work was published under.

NoDerivatives. You let people copy, distribute, display, and perform only verbatim copies of your work – not make derivative works based on it. If they want to alter, transform, build upon, or remix your work, they must contact you for permission.

Source: © Creative Commons, CC BY.

cannot use technological measures to restrict access to the work by others.⁷

Unlike commercial licenses, the Creative Commons licenses *allow* everything that is not restricted by the licensor or by the law. It is a very important principle to allow most use of the content, and to keep the content free and open. In commercial licenses, it is the other way around, where usually everything is forbidden that is not explicitly allowed.

When choosing a Creative Commons license, the licensor needs to choose between several options that can be combined with each other. These options are each represented by a graphical symbol. All Creative Commons licenses, except the CC0-license⁸, use the symbol  which stands for *Attribution*, and means that the licensee must attribute the work in the manner specified by the author or licensor⁹. All Creative Commons licenses allow the licensee to share the content.

The licensor needs to decide whether she or he wants to allow commercial use of the material. If she or he wants to restrict commercial use, the NC-property is specified by the graphical symbol  (also denoted  in Europe, and  in Japan). The next decision is whether derivative works are allowed. If derivative work is prohibited, marked with ND or the graphical symbol , the licensee may not alter, transform, or build upon this work other than what is defined under fair use or similar.

If a licensor decides to allow derivative works, she or he may also choose to require that anyone who uses the work to make that new work available under the same license terms, called *ShareAlike* and marked with SA or the graphical symbol . ShareAlike is a copyleft feature inspired by the GNU General Public License.

7. Note that the encrypted transfer or storage of CC-licensed content is allowed, as long as also an unencrypted version can be provided.

8. The CC0-license is explained in one of the next sections below.

9. One may not suggest in any way that the licensor endorses the licensee or the licensee's use of the work.



When combined, all the options that apply are collected as a combined logo, as shown to the left; alternatively the license is described by **CC BY**, followed by **-NC**, **-ND**, and/or **-SA**; in this example **CC BY-NC-SA**. Note that not all combinations of these options make sense; e.g., the no-derivative option (ND) together with share-alike (SA) is not viable. In principle, six different licenses, in addition to the CC0-license are possible.¹⁰

The CC licenses also state a limitation on liability. The license text states that *except to the extent required by applicable law, in no event will licensor be liable to you on any legal theory for any special, incidental, consequential, punitive or exemplary damages arising out of this license or the use of the work, even if licensor has been advised of the possibility of such damages.*

6.1.2 Implementation of the Creative Commons License

The Creative Commons public copyright licenses incorporate a *three-layer-design*: 1) Each license contains a traditional legal tool, in the kind of language and text formats that most lawyers understand, that is the *Legal Code Layer* of each license. 2) Since most creators, in fact, are not lawyers, the licenses are made available in a format that lay people can read – the *Commons Deed*, also known as the *human-readable* version of the license. The commons deed is a handy reference for licensors and licensees, summarising and expressing some of the most important terms and conditions. 3) The final layer of the license design provides the CC licenses in a *machine-readable* summary of the key freedoms and obligations. The CC have developed the *CC Rights Expression Language* (CC REL) to accomplish this.

6.1.3 Public Domain and the CC0 License

Copyright laws automatically apply copyright protection to works of authorship, irrespective of whether the author or creator wants those rights. The License CC0, graphically marked with , gives a creator a way to give up those rights to the fullest extent allowed by law.¹¹ Once the creator or a subsequent owner of a work applies CC0 to a work, the work is no longer his or hers in any meaningful sense under copyright law. Anyone can then use the work in any way and for any purpose, including commercial purposes. Thus, the CC0 can be considered as a “no rights reserved”-option. As the other CC licenses, the CC0 License is not revocable.

To apply CC0, the *affirmer* dedicates a work to the public domain by waiving all of his or her copyright and neighbouring and related rights in a work, to the fullest extent permitted by law. If the waiver is not effective for any reason, then CC0 acts as a license from the affirmer granting the public an unconditional, irrevocable, non-exclusive, royalty-free license to use the work for any purpose. CC0 is intended for use only by creators or holders of copyright and related or neighbouring rights (including *sui generis* database rights¹²), in connection with works that are still subject to those rights in one or more jurisdictions.

10. See <http://creativecommons.org/licenses/>; accessed August 11, 2011.

11. Applying the CC0 license does not absolve creators from other legal issues than copyright.

12. Database licenses and *sui generis* database rights are explained in Section 6.2.

terms of orig. work	Terms that can be used for a derivative work						
	BY	BY-NC	BY-NC-ND	BY-NC-SA	BY-ND	BY-SA	PD/CC0
PD/CC0	•	•	•	•	•	•	•
BY	•	•	•	•	•	•	
BY-NC		•	•	•			
BY-NC-ND							
BY-NC-SA				•			
BY-ND							
BY-SA						•	

Table 6.1. Compatibility chart for derivative work

The Public Domain Mark (PDM), denoted graphically as , differs from the CC0 in the way that PDM is intended for use with works that are already free of known copyright restrictions throughout the world, e.g., because the copyright is expired.

The CC0 and PDM tools also differ in terms of their effect when applied to a work. CC0 is legally operative in the sense that when it is applied, it changes the copyright status of the work, effectively relinquishing all copyright and related or neighbouring rights worldwide. In contrast, PDM is not legally operative in any respect – it is intended to function as a label, marking a work that is already free of known copyright restrictions.

Applying a CC license, other than CC0, to a work in the public domain may constitute *copyfraud* (Mazzone, 2006). However, incorporating a work that is in the public domain into a collection that itself is protected by copyright, then one may apply a Creative Commons license to the work as a collection. Including content into a collection does not affect the status of this work. Similarly, one may apply a Creative Commons license to an adaptation of a public domain work if one holds copyright to the adaptation.

6.1.4 Derivative Work and Collective Work

When using a Creative Commons-licensed work to create a *derivative work* or adaptation, the author of the derivative work is restricted in which license can be chosen. Table 6.1 shows which licenses may be chosen, given a CC license on the left. Notice that CC-licenses containing **ND** may not be the basis for derivative work. Note also that the CC licenses do not change, alter or modify fair use rights. Therefore, an author still may use fair use rights to incorporate CC works for any qualifying purpose.

Using CC-licensed material to create a collection, denoted as *collective work*, such as anthologies, encyclopedias and broadcasts, is allowed; however, the author of the collection needs to follow the original license. In practice, this means that material under any of the Creative Commons Non-commercial licenses cannot be included in a collection that is going to be used commercially. Note that when including a Creative Commons licensed works in a collection, *the work itself* must be kept under the original license. This doesn't mean the whole collection has to be put under this CC license – just the original work. For collective work it is important that the single parts are sufficiently separable from other parts of the collection.

6.1.5 Discussion

Creative Commons licenses are non-revocable. This means that one cannot stop someone, who has obtained work under a Creative Commons license, from using this work according to that license. The copyright holder can stop distributing the work under a Creative Commons license at any time; this will not, however, withdraw any copies of the work that already exist under a Creative Commons license from circulation, be they verbatim copies, copies included in collective works, or adaptations of this work. Therefore, authors should carefully consider this before releasing a work under a CC license.

The question how the Creative Commons licenses can foster innovation needs to be discussed. Creators can earn money from their work since the CC licenses are non-exclusive, and the creator is, therefore, not tied down to only make content available under the CC license. The creator can also enter into other revenue-generating efforts in relation to her or his work. The CC license can, for instance, be used to promote the creator's work.

The non-commercial license option is an inventive tool designed to allow people to maximise the distribution of their works while keeping control of the commercial aspects of their copyright. Here, the *non-commercial use* condition applies only to others who use the work, but not to the creator (the licensor). The non-commercial condition is therefore only imposed on the licensees, i.e., the users. People who want to copy or adapt works commercially under the non-commercial license must get the creator's permission first.

All jurisdictions allow some uses of copyrighted material without permission – such as quotation, current-affairs reporting, fair use, or parody – although these vary from country to country. These usage rights are independent from the license and are not affected or changed in any way. Thus, regardless of the jurisdiction a user is in, the CC licenses do not affect a user's right to use or allow use of content under copyright exceptions and limitations.

CC licenses are made available under royalty-free¹³ licenses. In the case of CC-licensed works that are licensed for non-commercial use only, the creator or licensor reserves the right to collect statutory royalties or royalties under compulsory licenses for commercial uses such as those collected for public performances; one may still have to pay a collecting society for such uses of CC-licensed works. However, these are indirect payments, not payments to the licensor.

When several licenses are applied to a work, only one of these is effective at a time, and the user can choose which. This applies also when using CC licenses. For example, if a work, e.g., a photograph, is governed by one license CC BY-NC, plus a separate license CC BY-ND, it does not mean that both provisions apply together. A user may, for instance, make derivatives of this work, but may not use these derivatives for commercial purposes; on the other hand, the user may sell the original image for commercial purposes. An owner who wants both provisions to apply together needs to choose one single license that contains both of these.

13. *Royalty-Free* refers to the right to use copyrighted material or intellectual property without the obligation to pay royalties to the licensor.

The use of a CC license is not recommended for software, hardware or databases. For software, instead, licenses made available by the Free Software Foundation or listed by the Open Source Initiative should be considered. Unlike the CC licenses, which do not make mention of source or object code, these existing licenses were designed specifically for use with software. Furthermore, the CC licenses are not compatible with the GPL. Note, however, that the CC0 Public Domain Dedication is GPL-compatible and acceptable for software. Note also, that the CC licenses are suited for software documentation, as for all text material. For databases and hardware applicable licensing regimes are discussed later in this chapter, in Sections 6.2 and 6.4.

A CC license terminates automatically if someone uses a work contrary to the license terms. This means that, if a user uses a work under a CC license and the user, e.g., fails to attribute a work in the specified manner, then this user no longer has the right to continue to use the work. This only applies in relation to the person in breach of the license; it does not apply generally to other people who use a work under a CC license and comply with its terms. A number of options can be used to enforce the terms, e.g., by contacting him or her to rectify the situation; or by consulting a lawyer to act on one's behalf.

In addition to the right of licensors to request removal of their name from a work when used in a derivative or collective they don't like, copyright laws in most jurisdictions around the world¹⁴ grant creators *moral rights* which may provide some redress if a derivative work represents a *derogatory treatment* of the licensor's work. Moral rights give an original author the right to object to *derogatory treatment* of their work; *derogatory treatment* is typically defined as *distortion or mutilation* of the work or treatment that is *prejudicial to the honour, or reputation of the author*. CC licenses do not affect any moral rights licensors may have¹⁵. This means that having moral rights as an original author of a work, a creator may be able to take action against a creator who is using a work in a way the creator finds objectionable. Of course, not all derivative works a creator does not like are necessarily *derogatory*.

6.1.6 Legal Considerations

As mentioned, all of the CC licenses contain a disclaimer of warranties, so there is no assurance whatsoever that the licensor has all the necessary rights to permit reuse of the licensed work. The disclaimer means that the licensor is not guaranteeing anything about the work, including that she owns the copyright to it, or that she has cleared any uses of third-party content that her work may be based on or incorporate.

This is typical of so-called *open source* licenses, where works are made widely and freely available for reuse at no charge. The original version 1.0 of the Creative Commons licenses contained a warranty, but the CC organisation ultimately concluded that, as with open source licenses, warranties and indemnities are best determined separately by private bargain, so that each licensor and licensee can determine the appropriate allocation of risk and reward for their unique situation. One option thus would be to use a private

14. with the notable exception of the US except in very limited circumstances.

15. with the exception of Canada.

contract to obtain a warranty and indemnification from the licensor, although it is likely that the licensor would charge for this.

As a result of the warranty disclaimer, before using a Creative Commons licensed work, creators should ensure that they have all the necessary rights to make the work available under a CC license. A user who is wrong in this assumption could be liable for copyright infringement based on use of the work. Additionally, CC licenses do not give permission to use any trademarks that may be associated with a CC-licensed work. In this case, the owner of a trademark needs to be asked for permission first.

6.1.7 Technical considerations

The Creative Commons offer the so-called partner interface that helps web-developers to license content in an open way. Using this interface for interactive content implies three steps: (1) letting users select a license by filling out a web form; (2) processing and storage of license information; and (3) display of license information. For other content, the Creative Commons offer diverse logos on their web site.

The Creative Commons Rights Expression Language (CC REL) is a specification for how license information may be described using the Resource Description Format (RDF) by the W3C (Beckett and McBride, 2004), and how license information may be attached to works. A rights expression language (REL) is a machine-processable language that expresses the rights one has in relation to content. A REL is a formal language, and differs from legal language in that it can be interpreted unambiguously by computers.

According to Abie (2009)¹⁶, DRM refers to the use of technologies which (1) unambiguously identify and describe digital information objects protected by intellectual property rights (IPR), (2) enforce fine-grained rules of usage for, and rights of access to, them, (3) monitor and track them, and (4) provide a secure infrastructure for their creation, distribution, storage, manipulation and communication, and finally (5) protect the privacy of users. While this definition addresses IPR in general, there are provisions in the copyright law that are difficult to enforce, such as the social and legal concepts of fair use. Currently, DRM is not conceived as an implementation of copyright law (González, 2005, p. 65). While copyright does not attempt to anticipate every possible use of a copyrighted work, DRM is based on allowing access according to specified rules. Note that copyright only addresses the use of content rather than to access to content. While the copyright law is an expression of “everything that is not forbidden is permitted”, DRM takes the approach of “everything that is not permitted is forbidden”.

Rights expression languages express IPR rules, such as the expression of copyright, and the expression of contract or license agreements. Also, it is a clear purpose of these expressions to control over access and use. A machine-actionable REL must use a formal, machine-readable language in order to be included in DRM.

González (2005) compares three different REL definitions according to their suitability

16. See Definition 5 in the book by Abie (2009).

for expressing copyright law: (1) the CC REL; (2) the ODRL REL (Iannella, 2002); and (3) the MPEG-21 REL (ISO, 2004). A similar comparison of four REL definitions is done by Coyle (2004).

The **The MPEG-21 REL** (ISO, 2004) is a part of the MPEG-21 standard (ISO 21000) (Burnett et al., 2006) that works in a trusted environment. The MPEG REL data model for rights expression consists of four basic entities and the relationship among these. This relationship is defined by the *assertion grant*, which consists of (a) the principal to whom the grant is issued; (b) the right that the grant specifies; (c) the resource to which the right in the grant applies; and (d) the condition that must be met before the right can be exercised. While the MPEG-21 REL is typical for languages that are based on the traditional IPR, Rodríguez and Delgado (2006) present how to achieve interoperability between MPEG-21 REL and the CC licenses.

The **The Open Digital Rights Language (ODRL)** by the W3C (Iannella, 2002) is a general-purpose language that allows, but does not require, actionable control over resource use. Iannella (2005) presents a draft of a Creative Commons profile for ODRL.

The **CC REL** is designed to describe the CC license in the terms of the copyright law. In contrast, MPEG-21 REL and ODRL are focused on the parties (e.g., issuer) to the license, but do not refer to copyright. The CC licenses in their machine-readable form tie the descriptions of *work* (as Dublin Core metadata elements) and *license* together. (González, 2005, Section 5.3) shows more technical details on the implementation of the CC REL.

6.1.8 Business Models for CC

In many ways, the business models for CC follow similar arguments as the business models for FOSS described in Section 4.7. Cross subsidisation and promotion are some of the major elements. As the CC are designed for creative works, some creators are not dependent on rights exclusion. Benkler (2007, p. 45) notes that creators can charge for the relationship rather than for the information.

In the classical business model for creative works, a rather small number of creators distribute via intermediaries (often using a copyright assignment) to as large an audience as possible. As costs for the distribution go down, and the *prosumer* enters the scene, there is less place for the intermediaries. Thus, new business models emerge using the CC licenses. The selection of the appropriate attribute, such as ND or NC is essential for the success, different in each single case.

Foong (2010) discusses examples from the film industry, and other creative areas how business models can be applied using the CC licenses. Feature films like *StarWreck* or *Cafuné* overcome limited exposure by using CC. These films are said to have innovated the film business. Other examples of CC business models include the music industry, where artists share their music using CC, but sell concert tickets and collector's items; lecturers share the slide show, but charge for a presentation; scientists use CC as a proof of their competence; and so on.

Connect with Fans (CwF). *Content as a product* gives way to *content as a service*, as Shirky notes. The direct and instantaneous nature of sharing content with fans over the Internet has the potential to create a sense of closeness between the creator and their fans (Foong, 2010). CwF needs to be combined with the *Reason to Buy* outlined in the following paragraph.

Reason to Buy (RtB). The RtB is a voluntary transaction, and a form of demand that is not artificially created by imposing legal scarcity on the work. Permission marketing is the privilege (not the right) of delivering anticipated, personal and relevant messages to people who actually want to get them (Godin, 2008). In this model, the creator, and the connection to the creator become the product; the relation to the creator becomes a value that cannot be substituted by the work that is distributed. Additionally, social pressure can enforce an RtB.

Services. Intermediaries, such as publishers, can offer services to the creators of CC, such as printing, distributing, copying, etc. Open Publishing is one business model, where the content is freely available while the publisher charges a moderate fee from the creator, and offers premium versions, such as a printed edition of the content.

6.2 Open Knowledge and Open Data

The term *open data* follows the idea that as much data as possible should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents, or other mechanisms of control.¹⁷ The access to data, stored in databases or otherwise made available, is becoming more important, especially for scientific data, with the purpose to benefit science; and public sector data, with the purpose to foster a more smoothly working information exchange for the citizens. New developments, e.g., the increased use of *apps* on many kinds of devices, make it necessary to access several data sources, and create a result that is presented to a user. Also the advent of *Linked Open Data* (LOD) has an important impact. In most jurisdictions there are intellectual property rights on data that prevent third-parties from using, reusing and redistributing data without explicit permission.¹⁸

Miller et al. (2008) point out that *copyright protection applies to acts of creativity, and categorically extends neither to databases nor those non-creative parts of their content*. While some individuals or organisations apply CC licenses to data, there is no meaningful legal basis to this.¹⁹ In some legislations, such as the US, the copyright can be applied to creative content, but not to databases. As counterexample, in the EU the Database Directive 96/9/EC

17. See http://en.wikipedia.org/wiki/Open_data; accessed August 17, 2011.

18. The text of this section is partially derived from *Open Definition* <http://www.opendefinition.org/guide/data/>; accessed August 15, 2011, and *Science Commons* <http://sciencecommons.org/resources/faq/databases>; accessed August 15, 2011, which are licensed CC BY.

19. This is one of the reasons why, for instance, OpenStreetMap is shifting license from CC BY-SA to the ODbL license explained later in this section.

Open data is often focussed on non-textual material such as maps, genomes, connectomes^a, chemical compounds, mathematical and scientific formulae, medical data and practice, bio-science and biodiversity. Problems often arise because these are commercially valuable or can be aggregated into works of value. Access to, or re-use of, the data is controlled by organisations, both public and private. Control may be through access restrictions, licenses, copyright, patents and charges for access or re-use. Advocates of open data argue that these restrictions are against the communal good and that these data should be made available without restriction or fee. In addition, it is important that the data are re-usable without requiring further permission, though the types of re-use (such as the creation of derivative works) may be controlled by license.^b

Source: © Wikipedia; licensed CC BY-SA.

a. A connectome is a comprehensive map of the human brain. See <http://en.wikipedia.org/wiki/Connectome>; accessed August 21, 2001.

b. See http://en.wikipedia.org/wiki/Open_data; accessed August 17, 2011.

creates a legal basis for copyright protection of databases. Miller et al. discuss these issues, present existing licenses as of 2008, and give a historic outline.

Databases usually are comprised of at least four elements: (1) a structure (or database model), which includes the organisation of fields and relations among them; (2) data sheets; (3) a set of field names identifying the data; and (4) data. All of the CC licenses can be applied to these elements to the extent that copyright applies to them. Copyright applies to minimally creative works expressed in a fixed form. In most databases, items (1) and (2) – the structure and the data sheet – will reflect sufficient creativity for copyright to apply. A CC license applied to these elements will permit copying of these elements under the conditions of the license selected.

There are three things to keep in mind when considering whether to apply a CC license to a database: (1) that the necessary rights or permissions have been obtained to make a database and any copyrightable elements are available under a CC license; (2) that only those parts of the database that the database provider wants to make available under a CC license are so licensed; and (3) if not all aspects of the database are protected by copyright, there should be a clear statement to this effect to indicate to users which aspects are subject to the license and which are not.

This distinction between the *contents* of a database and the database as a *collection* is especially crucial for factual databases since no jurisdiction grants a monopoly right on individual facts, i.e., the *contents*, even though it may grant right(s) to them as a collection. To illustrate, consider the simple example of a database which lists facts from natural science. While the database as a whole might be protected by law so that one is not allowed to access, reuse or redistribute it without permission this would never prevent anybody from stating a single fact stored in the database.

We should point out that barring any legal protection many providers of (closed) databases are able to use a simple contract (e.g., a EULA) combined with legal provisions prohibiting violation of access-control mechanisms to achieve similar results to a formal IP right,

With databases, there are likely four components to consider:

- (i) **The database model** is a specification describing how a database is structured and organised, including database tables and table indexes. The selection, coordination, and arrangement of the contents is subject to copyright if it is sufficiently original. The threshold of originality required for copyright is fairly low in many jurisdictions. [...] These determinations are very fact-specific and vary by jurisdiction.
- (ii) **The data entry & output sheets** contain questions, and the answers to these questions are stored in a database. For example, a web page asking a scientist to enter a gene's name, its pathway information, and its ontology would constitute a data entry sheet. The format and layout of these sheets are protected by copyright according to the same standard of originality used to analyse copyright in the database model.
- (iii) **Field names** describe data sets. For example, *address* might be the name of the field for street address information. These are less likely to be protected by copyright because they often do not reflect originality.
- (iv) **The data** contained in the database are subject to copyright if they are sufficiently creative. Original poems contained in a database would be protected by copyright, but purely factual data (such as gene names without more) contained in a database would not. Facts are not subject to copyright, nor are the ideas underlying copyrighted content.

Source: © Creative Commons, <http://wiki.creativecommons.org/Data>; accessed March 1, 2012 CC BY.

e.g., requiring users to log in with some credentials.

Forms of protection fall broadly into two cases: (1) Copyright for compilations; and (2) A *sui generis*²⁰ right for collections of data. However, there are no general rules and the situation varies by jurisdiction. The EU Database Directive 96/9/EC creates a legal basis for copyright protection of databases²¹. It is designed to let licensors explicitly use the copyright laws as a basis for licensing.

6.2.1 Open Data Commons

The *Open Data Commons* (ODC), created in December 2007 by Jordan Hatcher, is a project run by the *Open Knowledge Foundation*²² (OKFN). The OKFN defines open data, open content and open services²³. The *Open Knowledge Definition* (OKD) shows principles for licensing any kind of open content or data:²⁴ The OKD states:

A piece of content or data is open if anyone is free to use, reuse, and redistribute it – subject only, at most, to the requirement to attribute and share-alike.

20. *sui generis*: latin: of its own kind; unique in its characteristics; denotes an idea, an entity, or a reality which cannot be included in a wider concept.

21. The EU Database Directive 96/9/EC is available at http://eur-lex.europa.eu/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=en&numdoc=31996L0009&model=guichett; accessed August 15, 2011. See also <http://www.opendefinition.org/guide/data/>; accessed August 15, 2011.

22. See okfn.org; accessed August 15, 2011.

23. See opendefinition.org; accessed August 15, 2011.

24. Open Definition states that the OKD sets out principles to define openness in knowledge – that is any kind of content or data from sonnets to statistics, genes to geodata.

The OKFN also gives the *Open Software Service Definition* (OSSD) (Villa, 2007) that defines openness in relation to online (software) services

A service is open if its source code is Free/Open Source Software and non-personal data is open as in the OKD.

The OKFN lists a set of OKD-conformant licenses, and discusses *open government data and content*, *open data in science*, and *open bibliographic data* separately. For data and databases the ODC offers four different licenses²⁵ which have become a standard way to license open data:

PDDL. The Public Domain Dedication and License²⁶ places the data and/or the database in the public domain, which means waiving all rights.

ODC-BY. The Open Data Commons Attribution License²⁷ allows the user to share (copy, distribute and use the database), create (produce works from the database), and adapt (modify, transform and build upon the database), as long as proper attribution is given. The user must attribute any public use of the database, or works produced from the database, in the manner specified in the license. For any use or redistribution of the database, or works produced from it, the user must make clear to others the license of the database and keep intact any notices on the original database.

ODbL. The Open Database License²⁸, also denoted as *Attribution Share-Alike for data and databases*, allows the user to share, create, and adapt, as long as proper attribution is given in similar terms as for the ODC-BY license. Additionally, if the user publicly uses any adapted version of this database, or works produced from an adapted database, he or she must also offer that adapted database under the ODbL. If a user re-distributes the database, or an adapted version of it, then he or she may use technological measures that restrict the work, such as encryption or DRM, as long as he or she also redistributes a version without such measures. When creating or using a *produced work* publicly, a notice must be included with the produced work so that persons exposed to it are aware where the content was obtained from.

DbCL. The Database Contents License²⁹ waives all rights to the *individual contents* of a database licensed under the ODbL. The role of the DbCL is that data retrieved from a database can be used in an open database licenses, specifically the ODbL.

Note that the ODC licenses do not disallow commercial use; i.e., there is no ODC license with an NC-attribute. Likewise, there is no ODC license with an ND-attribute.

Besides the dimensions of *collective* and *derivative* databases the ODC licenses use the term *produced work* for work resulting from using the whole or a substantial part of the contents from a database, a derivative database, or a database as part of a collective

25. See <http://opendatacommons.org/faq/licenses/>; accessed August 15, 2011.

26. See <http://opendatacommons.org/licenses/pddl/>; accessed August 15, 2011.

27. See <http://opendatacommons.org/licenses/by/summary/>; accessed August 15, 2011.

28. See <http://opendatacommons.org/licenses/odbl/summary/>; accessed August 15, 2011.

29. See <http://opendatacommons.org/licenses/dbcl/>; accessed August 15, 2011.

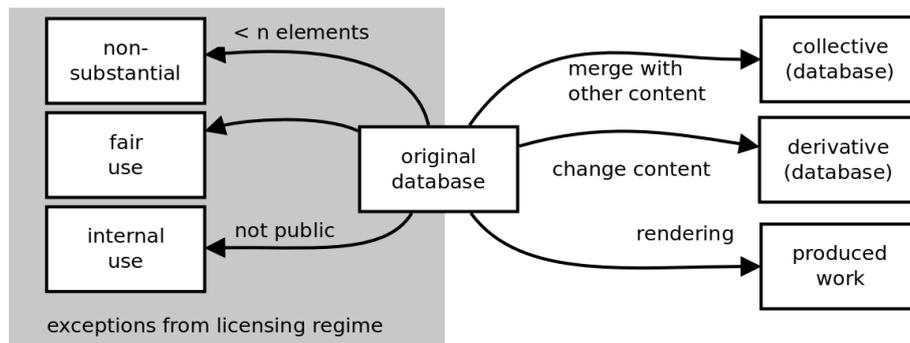


Figure 6.1. Different dimensions when conveying a database

database. Produced work can, e.g., be images, audiovisual material, text, or sounds. For produced work, the license notice only needs to be made if the produced work is used publicly. It is still under discussion whether attribution for produced work is required, as the current version of the license does. Note that also derivative databases used in the creation of publicly available produced works are subject to share-alike³⁰. The creation of a collective database does not require the collective database to be share-alike.

The term *convey* means in this context using a database, a derivative database, or a database as part of a collective database in any way that enables a person to make or receive copies of the database or derivative database. Note that conveying does not include interaction with a user through a computer network, or creating and using a produced work, where no transfer of a copy of the database or derivative database occurs.

In Figure 6.1 we show what happens when conveying data from a database: (1) conveying the data in the database are considered as a derivative database; (2) combining databases with different types of content will result in collective work; (3) rendering the content to a graph or an image will result in a produced work. Besides these three categories, we find (4) internal use, i.e., the content or the produced work is not public; (5) fair use, which is an exception from the copyright; and (6) non-substantial use of data, which means a non-repetitive and non-systematic access of very few elements for whatever purpose.

6.2.2 Closed Data and Restrictions to Openness

For the sake of completeness, we list intentional or unintentional mechanisms for restricting access to or re-use of data. These include (a) access control; (b) proprietary or closed technologies or encryption to create barriers for access; (c) copyright forbidding re-use of data; (d) licensing forbidding re-use of data; (e) patents forbidding re-use of data; (f) access restrictions for certain access, such as for search engines; (g) time-limited access such as subscription-based services; and (h) political, commercial or legal pressure.³¹ Provided that the owner of a service providing closed data is not committing copyfraud, the owner of such services is in his or her full right to offer data with the above restrictions, and close the data for commercial or other reasons. While this might be common practice for com-

30. See opendatacommons.org/news/; accessed August 19, 2011.

31. See http://en.wikipedia.org/wiki/Open_data; accessed August 17, 2011.

mercial entities, the data should be opened up by the service when there is a common public interest in services based upon these data.

As an example of data that contain licensing restrictions to openness, we mention the ASTER GDEM site which distributes geographic elevation data. While the data are freely available for download, the users must register. Derivatives of the data are allowed, but the derived data can only be distributed if the original data are not possible to reconstruct from the derived data set.³² Thus, simple format transformations are not allowed to be distributed while re-sampled data are.

6.2.3 Open Data in Science

Science is based on building on, reusing and openly criticising the published body of scientific knowledge. For science to effectively function, and for society to reap the full benefits from scientific endeavours, it is crucial that scientific data be made open³³, i.e., freely available on the public Internet permitting any user to download, copy, analyse, re-process, or use them for any other purpose without financial, legal, or technical barriers other than those inseparable from gaining access to the Internet itself. The Panton Principles³⁴ advocate that all scientific data should be explicitly placed in the public domain, but also embrace open licenses. They discourage licenses that limit commercial re-use or limit the production of derivative works by excluding use for particular purposes or by specific persons or organisations.

For scientific data, no separate licenses are necessary. However, the Panton Principles discuss the use of open data and their licenses, especially whether third-party data can be combined with open data, and released as open data. While they mention restrictions that may forbid this, they recommend to make a judgement whether data might be facts, whether it is likely to infringe “sui-generis” rights, and to adhere to community norms. We discuss some of the problems connected to mixing incompatible licenses below.

6.2.4 Linked Open Data

The term *linked data* describes a method of publishing structured data so that these can be interlinked and become more useful. Linked data builds upon the following four steps:³⁵ (i) use URIs as names for things; (ii) use HTTP URIs to be able to look up those names; (iii) when someone looks up a URI, provide useful information, using standards such as RDF (Beckett and McBride, 2004) or SPARQL (Prud’hommeaux and Seaborne, 2008); (iv) include links to other URIs which can be followed. In this way, the data of different databases are interlinked. While this principle works with all databases a user can access, the full potential can only be unleashed when there is open access to all the necessary data. If some data are closed, and thus unavailable to some users, a service using linked data can fail to provide high-quality results.

32. See <http://www.gdem.aster.ersdac.or.jp/faq.jsp>; accessed March 4, 2012.

33. Note that single facts are always open; however, a collection of facts may be protected, as may be observations from experiments.

34. See pantonprinciples.org; accessed August 24, 2011.

35. See http://en.wikipedia.org/wiki/Linked_data; accessed August 17, 2010.

Linked Open Data (LOD) combines open data with linked data, given proper license conditions. Access restriction as well as data license incompatibilities can affect the quality of a service using linked data. Combining data with LOD technology from many open databases³⁶, as used for science, is for the benefit of all.

Combining data in a derivative or produced work requires that the licenses for the combined data (mashup) are compatible if intended for public use. Specifically, the share-alike property can have an impact on whether a mashup of two databases is possible. This problem might even have an impact on the design of a system.

Tsiavos (2011) points out that there are two types of potential license incompatibilities:

(a) Incompatibilities due to different licensing terms, e.g., mashing incompatible CC licenses, or CC licenses and All Rights Reserved, or CC and ODbL in the wrong way; i.e., not as container (ODbL) and contained (CC) but rather as database with database. This may only be resolved by re-licensing the sources with compatible licenses. this effectively means taking new permissions by the original licensors.

(b) Incompatible due to other legal constrains, mainly personal data. This means, that before the data are anonymised or consent is obtained, they cannot be licensed. Anonymising or obtaining consent may lead to further problems with the data protection law, as the consent has to be specific, and opening up the data makes the consent very broad. This is actually not a matter of incompatibility but rather a specific legal problem which can only be solved if one adheres to the specific data protection laws of the jurisdictions where the original processing takes place.

Consider, for instance, a service on the Internet that creates a produced work, such as a chart, from open data licensed with a share-alike property (CC BY-SA), and personal data that are by law not allowed to be shared. This could be a relevant case for scientists who want a graphical presentation of their findings. From the discussion above we conclude that a derivative database cannot be created without first creating a dataset that contains no personal information, and that can be licensed with a compatible license to the open data. Note also, that this problem also affects produced work, since the current version of the ODbL license states this explicitly. If a produced work from the two data sources is not public, or if one uses the copyright exception of fair use, then the produced work can be created, provided that no external services are used to do the processing.

Tsiavos (2011) recommends to have (a) meta-data fields containing the license types; (b) license compatibility wizards³⁷; and (c) data protection compliance tools³⁸. This would at least show that one has taken all reasonable measures to avoid IPR infringements and data protection violations, though it would not indemnify or absolve the creator of all liability.

36. See <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>; accessed August 17, 2011.

37. such as the JISC www.web2rights.com/OERIPRSupport/creativecommons/; accessed August 24, 2011.

38. or at least a metadata field asking if data are personal and/or sensitive, and if they have been anonymised or consented.

6.2.5 Business Models

In many ways, the business models for data commons follow similar arguments as the business models for FOSS described in Section 4.7, and for the CC described in Section 6.1.8. Cross subsidisation and promotion of services are some of the major motivations for the data owners, while other enterprises can contribute with diverse services and application development around these data. Tammisto and Lindman (2011) present case studies for open data business models for businesses in Finland. They distinguish as business models for open data between *consulting*, *conversion*, *application*, and others.

Cross subsidisation. In many cases, the data owners are interested in distributing their data cost-effectively to as many potential customers as possible; e.g., time tables for (public) transport. Considering value also as non-monetary, the public sector can be considered a special case of cross subsidisation when citizens can access data freely. However, some types of data can only be accessible as a service, e.g., due to privacy reasons. Note that such data in the public sector previously have been licensed to enterprises who have used proprietary business models to distribute these data; of course, these enterprises oppose a change to open data licenses.

Presentation of Data. The development of web applications and mobile services to present (open) data (so-called *apps*) can be used as a business model by software developers. These *apps* often can be purchased for a rather small fee from an *app store*. Note the tight relations to the business models for software. Examples are *apps* for public transport, maps, weather, etc. Some of these *app* developers also include advertisement into their products; note, however, that selling advertisements in connection with open data is a business model for the *app* developer rather than the data owner.

Refinement and Processing. Some enterprises use open data and refine, convert, and process these either as a service for a paying customer or they offer refined data as a service. As an example, *geofabrik.de* offers consulting, training, and software development for OpenStreetMap, including activities to create data sets on specific subjects.

Linking together several data sets, transforming and processing these, especially in connection with linked open data (LOD) is one possibility that enterprises can offer as a service, in application development, or as consulting.

Public Data. Some private enterprises get paid to create and maintain data for the public sector, such as weather forecasts and measurements, maps, statistics, etc. While these data have been in the ownership of public institutions, the companies get a revenue both from the government, and from customers using their services and products. Opening up these data might threaten their source of income. The customers of the services might feel that they pay twice, via taxes and by paying for the services.

6.3 Governmental Licenses

The public sector needs to provide both citizens, enterprises, and others with information. In order for the society to work, citizens and decision makers need to be informed based on data made available. Without useful access to data, society eventually will suffer due to decisions based on wrong assumptions.³⁹ In many cases, public sector data is not open and free. Examples include that one has to pay a fee to access registered data for building applications, that one has to pay license fees to the organisation administering geodata when buying a map, or that meteorological data might be subject to license fees.⁴⁰

Many governments and public sector organisations want, for the benefit of a better working society, the access to information to be open and free. For instance, this is stated by the Norwegian Government in the Soria Moria II Declaration (Stoltenberg et al., 2009, p. 56):

*[The Government shall]
...see to it that information of public interest as a rule should be free, accessible and available for everyone in digital form*

Inspired by the blog entry by Lunde-Danbolt (2010) we present considerations on which elements an open license for public sector data should contain.

The copyright laws distinguish between (1) economical and (2) ideal rights to a work. The economical rights include the right to copy, and the right to decide whether it shall be made available to the public; in contrast, ideal rights include the right to be credited (attribution), and to be treated with respect: a work shall not be published in a way that violates a creator's or the work's reputation. This is also valid for public sector data. On the one hand, data should be freely available, while, on the other hand, the owner of the data wants to be credited, wants to assure that the data are not used to harm the owner's repudiation, and wants to assure that the data are only used in a lawful way.

In addition to economical conditions and ideal conditions, a data owner also might want to set certain terms of use, such as terms for access (e.g., registering for download), data freshness, or conditions for service quality.

Elements, such as (a) attribution, (b) no derivatives, and (c) share-alike are similar to the conditions of the CC licenses. Also the fact that the data owners accept (d) no liability, is similar to the CC licenses. Some of the data owners require (e) obligations to register any

39. A recent example in Norway tied to the service `fiksgatami.no` shows the importance of openly accessible data in the public sector. Reinholdtsen (2011) indicates in an email that requests to `fiksgatami.no` are forwarded to the wrong municipality since the borders between municipalities are not available to the underlying service based on data from OpenStreetMap with the necessary resolution. The correct data are owned by the Statens Kartverk who are currently not sharing these data under an open license. If these data were available to the `fiksgatami.no` service or its underlying services, wrongly re-directed requests to municipalities could have been avoided.

40. In Norway, the Statens Kartverk requires license fees for geodata while the personal access to interactive maps is granted without fees. Meteorological data are now freely available, e.g., through the web site `yr.no`.

An *orphan work* is a copyrighted work for which the copyright owner cannot be contacted. In some cases the name of the creator or copyright owner of an orphan work may be known, but other than the name no information can be established. Reasons for a work to be orphan include that the copyright owner is unaware of their ownership, or that the copyright owner has died or gone out of business, and it is not possible to establish to whom ownership of the copyright has passed.

Source: en.wikipedia.org/wiki/Orphan_works © Wikipedia, CC BY-SA.

Frame 6.2. Orphan works

download or use of data. Some data owners require (f) intermediate storage to avoid that their server infrastructure suffers from overload. With the increased use of ‘apps’ that download data, this problem has strongly increased lately⁴¹. Since the data users have certain expectations to data freshness, data owners can specify (g) terms for update, i.e., how often data need to be downloaded so that the application uses useful data. Also, data owners can specify (h) formats, service quality, up-time, etc. in their terms, including the mechanism for (i) versioning, if applicable. Note that terms (g) to (i) are *terms of service* rather than being an issue for licensing.

In practice today, many public sector data are made available without clear license or terms of use. Therefore, data cannot be used properly. Sometimes, unclear licenses are applied which are special-purpose, and often difficult to interpret legally.

Another growing problem for public sector data are *orphaned data*, where the copyright holder is unknown, or otherwise unavailable. The term *orphan works* is defined in Frame 6.2. These data need to be properly licensed in order to provide complete services. Note that data where the copyright has ceased are in the public domain. An example for these data are maps and geodata older than a certain number of years.

In the following, we discuss the UK *Open Government License* and the Norwegian *NLOD* in more detail. Other governments, such as the US-based *data.gov* also provide open data; however, it is rather difficult to find out which license they are using.

6.3.1 Open Government License

In the Open Government License (version 1.0)⁴² in the UK the licensor grants the licensee a worldwide, royalty-free, perpetual, non-exclusive license to use the information subject to the condition that the attribution statement specified by the information providers is included. The licensee is free to (i) copy, publish, distribute, and transmit the information; (ii) adapt the information; (iii) exploit the information commercially for example, by combining it with other information, or by including it in a product or application. The licensee must ensure that data are not used in a way that suggests any official status or

41. Solutions to this problem include download size limitations, obligation to register, exclude certain applications, etc.

42. See <http://www.nationalarchives.gov.uk/doc/open-government-licence/>; accessed August 15, 2011.

that the information provider endorses the licensee or his or her use of the information. It also must be ensured that neither the information nor its source are misrepresented, and that the data protection act, respectively the EU Privacy and Electronic Communications Regulations 2003, are not breached. In case the licensee fails to comply to these conditions the license will automatically end.

The Open Government License does not cover the use of (i) personal data in the information; (ii) information that has neither been published nor disclosed under information access legislation; (iii) departmental or public sector organisation logos, crests, etc.; (iv) military insignia; (v) third party rights the information provider is not authorised to license; (vi) information subject to other intellectual property rights, including patents, trademarks, and design rights; and (vii) identity documents, such as passports and drivers licenses.

6.3.2 NLOD

The *Norsk Lisens for Offentlige Data* (*engl.* Norwegian License for Public Data, NLOD)⁴³ has been in a hearing phase by the *Fornyingsdepartementet* (*engl.* Norwegian Ministry of Government Administration, Reform and Church Affairs) (Lunde-Danbolt, 2011). Lunde-Danbolt (2010) gives considerations made while creating the NLOD, while comments during the hearing phase are available (nlo, 2011).

While the creators of the license envisioned the possibility to choose between a variety of licenses⁴⁴, the license that is currently in a hearing process is similar to, and compatible with the CC BY license. As of March 2012, data licensed with NLOD include diverse statistics data, data from public transportation and aviation, traffic information for roads, data from libraries, weather data, prices for electricity, and results from municipal elections.⁴⁵

6.4 Hardware Licenses

Open Hardware, also open source hardware⁴⁶⁴⁷ (OSHW) consists of physical artifacts of technology designed and offered in an open way. OSHW has many similarities with FOSS. The term OSHW is usually applied to information about hardware design, such as mechanical drawings, schematics, bill of materials, source code in a hardware description language, printed or integrated circuit layout data, in addition to the software that drives the hardware.

Rather than creating a new license, some open source hardware projects use existing, open source software licenses, such as the BSD, GPL and LGPL licenses. However, de-

43. See <http://data.norge.no/nlod/>; accessed August 15, 2011; an annotated version is available at <http://data.norge.no/nlod/annotert-lisens/>; accessed August 15, 2011; in Norwegian only.

44. denoted as “clause buffet”.

45. See <http://data.norge.no/data/>; accessed March 1, 2012.

46. See http://en.wikipedia.org/wiki/Open-source_hardware; accessed August 22, 2011.

47. The principles and definition of OSHW is given at freedomdefined.org/OSHW; accessed August 23, 2011. This definition is similar to the definition of FOSS, except some adaptations that are specific to hardware.

spite superficial similarities to software licenses, most hardware licenses are fundamentally different for various reasons: (a) The final product is a material good that cannot be copied with nearly zero costs, as is the case with software. Therefore, the license can only be applied to the design rather than to the final product. (b) The design and the documentation can be considered as the “source code” of the hardware. To the design and to the documentation software and content licenses could be applied. However, using content licenses, the relationship between a document and the resulting hardware cannot be expressed. Using software licenses, not all terms really make sense, while other terms are undefined. (c) By nature, hardware licenses typically rely more on patent law than on copyright law. Whereas a copyright license may control the distribution of the source code or design documents, a patent license may control the use and manufacturing of the physical device built from the design documents.

McNamara (2007) defines four possible levels of openness in open hardware projects: (1) *closed*: any hardware for which its creator does not release any information; (2) *open interface*: documentation on how to make a piece of hardware perform its designed function is available (minimum level of openness); (3) *open design*: documentation is provided so that a functionally compatible device could be created by a third party; (4) *open implementation*: additionally, the complete bill of materials necessary to construct the device is available.

6.4.1 The TAPR Open Hardware License

Ackermann (2009) presents the motivation for open source hardware licenses based on the design process for hardware. He develops the Tucson Amateur Packet Radio Corporation⁴⁸ (TAPR) Open Hardware License⁴⁹ (OHL). The TAPR OHL is designed in the spirit of the GNU GPL, but the OHL is not primarily a copyright license. While copyright protects documents, software, and data from unauthorised copying, modification, and distribution, it does not apply to make, distribute or use a hardware design based on these documents. Although the OHL does not prohibit anyone from patenting inventions embodied in an open hardware design, and cannot prevent a third party from enforcing their patent rights, those who benefit from a design licensed under the OHL may not bring lawsuits claiming that this design infringes their patents or other intellectual property. Note that the OHL addresses the issues of creating tangible, physical things, but does not cover software, firmware, or code loaded into programmable devices, for which the GPL suits better.

The OHL states in its preamble that a licensee can modify the documentation and make products based upon it. These products may be used for any legal purpose without limitation. Such products may be distributed to third parties if the respective documentation is made available to anyone who requests it for at least three years. The unmodified documentation may be distributed only as the complete package as received. Modified documentation or products based on it may be distributed under the OHL license (share-

48. See www.tapr.org; accessed August 23, 2011.

49. See www.tapr.org/ohl.html; accessed August 23, 2011.

alike). Additionally, all previous developers who have stated their email address need to be informed about the new changes according to rules stated in the license. Making documents available to others includes the requirement that both the previous version, as well as the changed version need to be included, as well as a text file that describes the changes.

The OHL also addresses that patents or registered designs held by the licensor can be used by the licensee to the extent necessary. Note, however, that the licensor cannot grant rights for patents or registered designs he or she does not own.

Some of these requirements are different from software licenses. The requirement to inform the previous creators explicitly, and the requirement to include both the “before” and “after” versions are specific to the TAPR OHL.

According to Paul (2007) the Open Source Initiative (OSI) with its president Eric S. Raymond expressed some concern about certain aspects of the OHL, since the term “distribution” is differently interpreted in some parts.

6.4.2 The CERN Open Hardware License

The CERN OHL⁵⁰ is a recent open hardware license. The terms of the CERN OHL are similar to the TAPR OHL.

6.4.3 Business Models

Menichinelli (2011) presents business models for open hardware.⁵¹ According to Ferreira and Tanev (2009) there is little specific research on open hardware business models available. They examined four companies, 88 market offers and 93 open hardware projects in order to identify seven business models. With this, they extend the list of four business models presented by Salem and Khatib (2004). The business models presented by Ferreira and Tanev include: (1) services, such as customisation, expertise, and consulting over owned or third party open hardware; (2) manufacturing of owned or third party open hardware; (3) manufacturing of proprietary hardware based on open hardware; (4) dual licensing (as in FOSS); (5) proprietary hardware designs based on open hardware; (6) hardware tools, e.g., development boards for testing and verification, for open hardware; and (7) proprietary software tools for developing open hardware.

6.5 Equity-based Licenses

The *equity-based licenses*⁵² are approaches that aim to radicalise the existing mainstream copyleft approaches such as the CC SA and the GNU GPL approach, in the sense of more equity. In these approaches, ownership, as well as economic, ethical and democratic

50. See <http://www.ohwr.org/cernohl>; accessed August 23, 2011.

51. See also the presentation of open hardware business models by David Rowe at <https://fossbazaar.org/content/david-rowe-open-hardware-business-models>; accessed February 1, 2012, presented at linux.conf.2009.

52. See http://p2pfoundation.net/Equity-based_Licenses; accessed February 19, 2012. The term *equity* refers to fairness; see <http://en.wikipedia.org/wiki/Equity>; accessed March 1, 2012.

rights and duties are introduced in addition to existing licensing schemes. Though using share-alike, the equity-based licenses are not compatible with the CC SA licenses.

6.5.1 Peer Production License

The Peer Production License (Magyar and Kleiner, 2010), derived from the CC BY-SA-NC license, restricts commercial exploitation of works only for worker-owned businesses or worker owned collectives where all financial gain, surplus, profits and benefits produced by the business or collective are distributed among the worker-owners. This means that enterprises need a certain form of organisation, e.g., being organised as a cooperative, in order to use the Peer Production License. Kleiner (2010) rejects the Creative Commons since he claims that the pick-and-choose licenses of the CC allow arbitrary restrictions by the authors, rather than keeping content free.⁵³ Though derived from the CC BY-SA-NC license, the peer-production license is not compatible with the CC SA licenses.

As a rationale behind the Peer Production License, Kleiner claims that *while copyleft is very effective in creating a commons of software, to achieve a commons of cultural works requires copyleft, a form of free licensing that denies free access to organisations that hold their own assets outside the commons*. One reason for this distinction is that artwork is consumer demand rather than capital demand, like for software. Kleiner concludes that copyleft must become *copyleft* which insists on workers right to own means of production.

6.5.2 IANG License

The IANG license⁵⁴ by Patrick Godeau is a free license for any type of intellectual creation that allows users to use, analyse, modify and distribute the creation to which it applies. It attributes economic rights in that it guarantees to everyone the freedom to access the accounting of each commercial distribution of the creation, and entrusts its economic management to those who finance it by their donations, purchases or investments. As a democratic exercise, it allows every person contributing to a creative or economic project based on the creation to participate in decisions concerning this project.

6.5.3 Genero License

The Genero project⁵⁵ is an effort to establish an ecosystem for production, reproduction and distribution of creative works with free licenses. It is a network of service providers connected to a federated registry that registers all free culture worldwide. The Genero license consists of a commercial addition to the CC licensing scheme, using the CC+ framework. Instead of providing specific commercial terms, the Genero license is a basic framework that enables any kind of business model. Typical business terms are unit cost and revenue share. The Genero license permits mashups and re-use without prior

53. According to Kleiner, Richard Stallman has made similar statements rejecting the CC.

54. See <http://iang.info/en/license.html>; accessed February 19, 2012, translated from <http://iang.info/fr/license.html>; accessed February 19, 2012; see also <http://iang.info/fr/manifesto.html>; accessed February 19, 2012.

55. See http://p2pfoundation.net/Genero_Initiative; accessed February 19, 2012.

permission, as long as parent works gets a fair share of any revenues.

6.5.4 Common Goods Public License

The Common Goods Public License (CGPL)⁵⁶ claims to be an ethical license based on the copyleft. It includes duties to humanity and the environment.

6.5.5 Business models

Inspired by communist theory, Kleiner (2010) argues that today's Internet and the *Web 2.0* follows a client-server architecture, serving only few large companies who own both infrastructure and content produced by individuals. Thus, the production means are not in the hands of the producers of content⁵⁷, and this is why most producers cannot retrieve monetary value from their works. *Venture Communism* provides a structure for independent producers to share a common stock of productive assets. Ownership in a venture commune can only be earned by labour alone, not land nor capital.

Venture Communism. The Peer Production License provides a business model only for worker-owned businesses or worker owned collectives through *venture communism*. Legally, a venture commune is a firm as a federation of workers' collectives and individual workers. Shares in the venture commune can only be earned by labour. Property necessary for the production is funded by bonds that are handled by certain rules that assure collective ownership.

Collection collective. This business model includes collecting and distributing revenue from works, similar to a collection society⁵⁸; both the IANG and the Genero licenses are suitable for this business model.

6.5.6 User Ownership Approach

Patrick Anderson promotes the *User Ownership Approach*⁵⁹ and Inter-Owner Trade Agreements (IOTA), for which the GNU General Public License is an example. He introduces the *GNU Public Law* that relies upon initial investing owners (developers) of physical or virtual objects. These owners can add a constraint to these aforementioned objects where all profit must be treated as an investment in more physical sources for the future production of the same kind of object. Wages are in this framework costs, i.e., payments for work as arranged between current owners and potential workers. According to Anderson, the GNU Public Law can be seen as a generalisation of the GNU General Public License.

The GNU General Public Law⁶⁰ is a generalisation of the goals of the FSF's GNU General

56. See <http://www.cgpl.org/index.php/the-license.html>; accessed February 20, 2012. The CGPL was published November 20, 2003.

57. Kleiner claims that the producers of content are the working class of the Internet.

58. See http://en.wikipedia.org/wiki/Collection_society; accessed February 20, 2012.

59. See http://p2pfoundation.net/User_Ownership; accessed February 19, 2012.

60. See http://p2pfoundation.net/General_Public_Law; accessed February 19, 2012.

Public License into the realm of the physical, tangible, and material world. It allows the use of an object for any purpose, modifying it by renting or buying the necessary physical sources for that modification, copying it by renting or buying the physical sources needed for that production, and to share it or a copy both as original and modified. All profit gained through the sale of an object needs to be treated as an investment from the user toward more physical sources. The concept also introduces a specific currency, the GNUrho, for this purpose.

6.6 Case Study: Licensing in OpenStreetMap

In this section we look into specific licensing issues in OpenStreetMap (OSM). As outlined in Section 2.3, OpenStreetMap represents a database of geodata, several types of map rendering, and documentation presented on a wiki. According to the OSM web pages⁶¹ OpenStreetMap consists of *open data*, licensed under the Creative Commons Attribution-ShareAlike 2.0 license (CC-BY-SA). Contributors to OSM can also choose to contribute their data into the public domain. The content of the OpenStreetMap Wiki is licensed CC BY-SA.

There is currently a license change ongoing⁶² for the data in OpenStreetMap from CC BY-SA 2.0 to ODbL 1.0. The reason for this license change is that the CC BY-SA license is not specifically designed for data bases. As of May 2010 all new users automatically accept the ODbL license; as of April 17, 2011 there is a mandatory accept or decline on the ODbL license for all users who contribute. In the case of accept this user's data are re-licensed to ODbL, while users who decline are not allowed to contribute any more. Their contributions will be removed from the data base when the license change will be performed. Anonymous contributions are no longer allowed.

The announced license change has been heavily disputed by the members of the OpenStreetMap community, as can be seen in the OpenStreetMap wiki⁶³. In order to evaluate the consequences to change to the ODbL license, a number of typical use cases of OpenStreetMap data⁶⁴ has been prepared, which have been analysed by OSM-members with legal expertise.

In OSM one distinguishes between data that are licensed, i.e., the content of the database, and the produced work, i.e., renderings of the data into maps. It was intended that there should be no license restriction on produced work other than not allowing reverse-engineering from produced works, i.e., creating a database from produced work. However, the release v1.0 of the ODbL did not approve this. Note that when publishing produced works, notice must be given that makes the user aware of where to obtain the database(s) the work is produced from.

61. See www.openstreetmap.org/copyright; accessed August 18, 2011.

62. As of spring 2012, the announced license change has not been performed, and the OSM data still are licensed CC BY-SA. However, the license change is announced to happen during 2012.

63. See wiki.openstreetmap.org/wiki/Open_Data_License/Why_You_Should_Decline; accessed August 18, 2011.

64. See wiki.openstreetmap.org/wiki/Open_Data_License/Use_Cases; accessed August 18, 2011.

In the comments to the above mentioned use cases several issues regarding derived, collective, and produced work are discussed. Overlaying maps with information from other databases or sources are considered collective work, but require the notice from where to obtain the database. However, when overlaying with confidential data it must be considered whether private or public use is applicable. When mashing up data, it must be considered whether the outcome is derived work or collective work. Here, community guidelines are required. Note that screen shots of produced work in most cases are considered fair use.

Encrypted databases, e.g., for use in games, are allowed as long as the derivative database also is offered unencrypted. Providing the OSM data in a proprietary format is allowed as long as also a non-proprietary version of the same data is offered. Using OSM data to fill up gaps in a proprietary data base in a commercial product without contributing these data is not allowed. Also frequent non-substantial extracts are considered a breach of the license.

References

- Høring - Norsk Lisens for Offentlige Data (NLOD), Høringsuttalelser. collection, web, 2011. URL <http://www.regjeringen.no/nb/dep/fad/dok/horinger/horingsdokumenter/2011/horing-nlod/horingsuttalelser.html?id=640253>. accessed February 19, 2012.
- Habtam Abie. *Distributed Digital Rights Management: Frameworks, Processes, Procedures and Algorithms*. VDM Verlag, October 2009. ISBN 3639202961.
- John R. Ackermann. Toward open source hardware. *Dayton L. Rev.* 183, 34(2):183–222, 2009.
- Dave Beckett and Brian McBride. RDF/XML syntax specification. W3C recommendation, W3C, February 2004. URL <http://www.w3.org/TR/REC-rdf-syntax/>.
- Yochai Benkler. *The Wealth of Networks*. Yale University Press, October 2007. ISBN 0300125771, 9780300125771.
- Ian S. Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen, editors. *The MPEG-21 Book*. John Wiley & Sons, 2006. ISBN 0-47001011-8.
- Karen Coyle. Rights expression languages. Report, Library of Congress, February 2004. URL www.loc.gov/standards/relreport.pdf.
- Edy Ferreira and Stoyan Tanev. How companies make money through involvement in open source hardware projects. *Open Source Business Resource*, Commercialization, February 2009. URL <http://www.osbr.ca/ojs/index.php/osbr/article/view/827/800>. accessed February 1, 2012.
- Cheryl Foong. Sharing with Creative Commons : a business model for content creators. *Platform : Journal of Media and Communication*, Creative Commons Special Edition:64–93, December 2010. ISSN 1836-5132.

- Seth Godin. Permission marketing. Seth Godin's blog, January 31 2008. URL http://sethgodin.typepad.com/seths_blog/2008/01/permission-mark.html. accessed January 27, 2012.
- Roberto García González. *A Semantic Web approach to Digital Rights Management*. PhD thesis, Universitat Pompeu Fabra, Barcelona, November 2005. URL rhizomik.net/html/~roberto/thesis/. Section 5: Rights Expression Languages.
- Renato Iannella. Open Digital Rights Language (ODRL), version 1.1. W3c note, World Wide Web Consortium, 2002. URL <http://www.w3.org/TR/odrl>.
- Renato Iannella. ODRL creative commons profile. W3c draft specification, World Wide Web Consortium, June 2005. URL <http://odrl.net/Profiles/CC/SPEC.html>.
- ISO. ISO/IEC 15938-5: 2003. information technology – multimedia framework (MPEG-21) Part 5: Rights Expression Language. International Organization for Standardization, 2004.
- Dmytri Kleiner. The Telekommunist Manifesto. Network Notebooks 03, Institute of Network Cultures, Amsterdam, 2010.
- Sverre Andreas Lunde-Danbolt. En klausulbuffet av vilkår. Fornyingsdepartementets blogg om viderebruk av offentlige data, august 2010. URL <http://data.norge.no/blogg/2010/08/en-klausulbuffet-av-vilkar/>. in Norwegian.
- Sverre Andreas Lunde-Danbolt. Norsk lisens for offentlige data (NLOD) skal på høring. Fornyingsdepartementets blogg om viderebruk av offentlige data, march 2011. URL <http://data.norge.no/blogg/2011/03/norsk-lisens-for-offentlige-data-nlod-skal-pa-horing/>. in Norwegian.
- John Magyar and Dmytri Kleiner. Peer production license: A model for copyfarleft. In Dmytri Kleiner, editor, *The Telekommunist Manifesto*, number 03 in Network Notebooks. Institute of Network Cultures, Amsterdam, 2010. ISBN 978-90-816021-2-9.
- Jason Mazzone. Copyfraud. *New York University Law Review*, 81:1026–1100, 2006. Brooklyn Law School, Legal Studies Paper No. 40.
- Patrick McNamara. Open hardware. *Open Source Business Resource*, Defining Open Source, September 2007. URL <http://www.osbr.ca/ojs/index.php/osbr/article/view/379/340>. accessed February 1, 2012.
- Massimo Menichinelli. Business models for open hardware. [openp2pdesign.org](http://www.openp2pdesign.org), March 16 2011. URL <http://www.openp2pdesign.org/2011/open-design/business-models-for-open-hardware/>. accessed January 30, 2012.
- Paul Miller, Rob Styles, and Tom Heath. Open data commons, a license for open data. *Proc. LDOW2008, April 22, 2008, Beijing, China*, April 2008. URL <http://events.linkeddata.org/ldow2008/papers/08-miller-styles-open-data-commons.pdf>.

- Ryan Paul. TAPR introduces open-source hardware license, OSI skeptical. blog post, 2007. URL arstechnica.com/old/content/2007/02/8911.ars. accessed August 23, 2011.
- Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. URL <http://www.w3.org/TR/rdf-sparql-query/>.
- Petter Reinholdtsen. Noen som kan forbedre kommunegrensen mellom Nøtterøy og Tønsberg? email to kart@nuug.no, August 21, 2011. URL <http://lists.nuug.no/pipermail/kart/2011-August/002948.html>. in Norwegian.
- Eva Rodríguez and Jaime Delgado. Towards the interoperability between MPEG-21 REL and Creative Commons Licenses. In *AXMEDIS'06*, pages 45–52, 2006.
- Mohamed A. Salem and Jamil I. Khatib. An introduction to open-source hardware development. *Eetimes News & Analysis*, 2004. URL <http://eetimes.com/electronics-news/4155052/>. accessed February 1, 2012.
- Clay Shirky. Help, the price of information has fallen and it can't get up. *Clay Shirky's Writings About the Internet*. URL http://www.shirky.com/writings/information_price.html. accessed January 27, 2012.
- Jens Stoltenberg, Kristin Halvorsen, Liv Signe Navarsete, Helga Pedersen, Audun Lysbakken, Lars Peder Brekk, Raymond Johansen, Bård Vegar Solhjell, and Trygve Slagsvold Vedum. Political platform as basis for the government's work formed by the labour party, socialists left party and centre party, 2009–2013. *Government Declaration, 2009*. URL <http://www.regjeringen.no/upload/SMK/Vedlegg/Rapporter/Plattform-sm2-a4-web-english.pdf>.
- Yulia Tammisto and Juho Lindman. Open data business models. In *proc. IRIS34, The 34th Information Systems Research Seminar in Scandinavia*. Turku School of Economics, August 16-19 2011.
- Prodromos Tsiavos. Discussing licensing problems. personal communication, August 18 2011.
- Luis Villa. evaluating a Free/Open Service Definition (rough draft). blog post, July 2007. URL tieguy.org/blog/2007/22/. last accessed: August 15, 2011.

7 Open Standards



by Wolfgang Leister

A *technical standard* is an established norm or requirement about technical systems. It is usually a formal document that establishes uniform engineering or technical criteria, methods, processes and practises. In contrast, a custom, convention, company product, corporate standard, etc. which becomes generally accepted is often called a *de facto standard*.¹ A technical standard is usually developed and maintained by an organisation, trade union, or a regulatory body². There is no legal limitation to who can develop and maintain standards, i.e., everybody can issue a standard³. Technical standards are usually designed as reference to technical requirement documents and contracts, and to foster interoperability between technical systems. However, as we will see later in this chapter, some organisations use standards to control innovation and marketshare.

Several types of technical standards are available: (a) A *standard specification* is a set of requirements for an item, material, component, system or service. In information technology, this might be the specification of a document or media format, such as text, images, or sound.⁴ (b) A *standard test method* defines procedures and metrics how to produce test results.⁵ Other, more informal technical standards include (c) *standard practise* as a set of instructions for performing operations or functions; and (d) *standard guide* as general information on a subject. Further, we mention the (e) *standard definition* as formally established technology; and (f) *standard units* from physics, chemistry and mathematics.

A *profile* to a standard is a selection of capabilities and specification of some parameters defined in a technical standard applicable to certain purposes. Since standards are defined as general as possible, some combinations of parameters are not applicable for some application areas. For example, while a standard for encoding video is not application-specific, profiles with specifications of parameter ranges and capabilities may be defined

1. See en.wikipedia.org/wiki/Technical_standard; accessed August 25, 2011.

2. Examples for organisations maintaining standardisation documents include the Internet Engineering Task Force (IETF), the World Wide Web Consortium (W3C), and the European Broadcasting Union (EBU). Examples for regulatory bodies are the International Organisation for Standardization (ISO), Deutsches Institut für Normung (German Institute for Standardisation, DIN), Standard Norge (SN), the International Telecommunication Union (ITU, an organisation under the United Nations) Telecommunication Standardization Sector (ITU-T), and the International Electrotechnical Commission (IEC).

3. Note that standards that are not adopted or supported are not worth much.

4. Examples for such standards include the Portable Document Format (PDF, ISO 32000-1), the image format JPEG (ISO/IEC 10918-1), or MPEG-1 (ISO/IEC 11172-1) and MPEG-2 (ISO/IEC 13818-1).

5. Examples for such standards include the ITU-T BT.700, a standard for video quality assessment, and standards how to test technical broadcast quality by the European Broadcasting Union (EBU, ebu.ch).

for its use for television, for the Internet, for mobile devices, and so forth. Therefore, when specifying compliance with a standard, it must be specified which profiles are applicable for a specific implementation.

The availability of standards differ depending on the publisher and the type of standard. For public documents, these are available in libraries or can be purchased for a fee, such as the standards by ISO. Other standards are freely accessible on the Internet, such as the standards by the W3C and the IETF. Standards by private bodies are circulated according to their own determination.

Some standards, such as standards for media coding⁶ in IT, only describe how to decode media content, while methods how to encode media need to be developed. However, most of these standards often contain a part denominated as *Reference Software* or similar, which contains an unoptimised implementation as a proof of concept.

Some standards contain information that relates to patents. The users of a standard, e.g., issued by the ISO, must be aware that these can contain patents, and are, therefore, not possible to implement without consent of the patent holder. One example is the video codec H.264.

7.1 Open Standards

The term *open standard* is defined differently by various organisations and scholars. The definitions contain various aspects, such as (1) publicly available and possible to copy, distribute and use freely or for a nominal fee; (2) free to use⁷; (3) implementable on royalty-free basis; (4) non-discriminatory with respect to who uses it and to what purpose; (5) non-discriminatory and reasonable fees for use; (6) open process regarding during definition of a standard; and (7) having a complete implementation available with an open license⁸.

Considering the term *open* as positive, the different organisations embrace different aspects of openness. In most organisations, a standard is approved by formalised committees according to a predefined voting process. In an open process, all parties that are interested can participate, and a consensus between these participants will define the standard. In some standardisation bodies, there are rules who can participate. For instance, the ISO allows a certain number of participants from every interested country that are appointed by the national standardisation bodies. In other organisations, such as the IETF, the participation is much wider, and more adapted to the CBPP principle.

Regarding royalties and patents, the W3C ensures that its specifications can be implemented on a royalty-free basis. On the other hand, major standardisation bodies, such as the IETF, the ISO, many national standardisation bodies, the IEC, and the ITU-T, permit that standards contain specifications whose implementation may require payment of

6. Examples include the standards mentioned in footnote 4 on the preceding page.

7. The term *use* in this context does not imply that the standardisation document is free of cost.

8. In the case of an IT standard, a FOSS implementation must be available; in the case of hardware, an implementation using an open hardware license must be available.

licensing fees, e.g., due to patents. Their definition of openness permits that the patent holder can impose *reasonable and non-discriminatory* royalty fees and other licensing terms in implementers and users of a standard.⁹

For the ITU-T, openness consists of a standard being *made available to the general public and [...] developed (or approved) and maintained via a collaborative and consensus driven process*, which is reasonably open to all interested parties. Intellectual property rights are licensed world-wide on a non-discriminatory basis to reasonable terms and conditions. The negotiations about these are left to the parties. The IETF operates with a similar definition.

The European Union requires open standards to be adopted and maintained by non-profit organisations, having the standardisation documents available freely or at a nominal fee, and not allowing constraints on the re-use of a standard. They require that intellectual property rights are made irrevocably available on a royalty-free basis. There are also many national definitions of what an open standard is.¹⁰

As outlined above, international standards from some standardisation bodies may contain intellectual property rights, such as patents. Creating FOSS based on technologies that build upon these standards might therefore be difficult, depending on the licensing terms. While the standardisation bodies talk about *reasonable and non-discriminatory royalties*, the issue of royalties is a unsolvable problem in FOSS. As a consequence, some parts of some standards may not be implemented as FOSS.¹¹

In order to avoid patents in standards, Perens (no date) defined six principles for standards based on (1) availability; (2) end-user choice; (3) no royalty; (4) no discrimination; (5) openness on extension or subset; and (6) limitations of predatory practises.

The W3C uses a different definition¹² based on 1) **transparency** – process in public; availability of technical discussions, meeting minutes, etc. 2) **relevance** – thorough analysis before starting standardisation; 3) **openness** – anybody can participate on a worldwide scale; 4) **impartiality and consensus**; 5) **availability** – free access to standard text; clear intellectual property rights rules for implementation, allowing FOSS development; and 6) **maintenance** – ongoing process for testing; revision; permanent access.

The organisation DIGISTAN¹³ defines open standards from the perspective of freedom

9. Among the here mentioned standardisation bodies, the IETF and the ITU-T name their standards as *open standards*. even though they contain a patent fee licensing requirement.

10. See en.wikipedia.org/wiki/Open_standard; accessed August 25, 2011.

11. An example: The well-known MP3 codec used in the music industry is part of MPEG-1 (ISO/IEC 11172-3:1993) as *MPEG Audio Layer 3*. The Fraunhofer IIS and Thomson Consumer Electronics have been granted patent rights on the MP3-technology, and they demand royalties on every distributed MP3-encoder, even if distributed as FOSS. The *BladeEnc* project that developed an MP3-encoder faced this problem. The software is licensed under the GNU GPL, but is not allowed to be downloaded or used in some jurisdictions; see <http://www2.arnes.si/~mmilut/>. As a consequence, the FOSS developers adopted alternative sets of codecs that do not contain patents, such as the Vorbis video codec, and related FOSS implementations; see <http://en.wikipedia.org/wiki/Vorbis>; accessed August 27, 2011.

12. See <http://www.w3.org/2005/09/dd-osd.html>; accessed August 27, 2011.

13. See <http://www.digistan.org/text:rationale>; accessed August 27, 2011.

Open Standards Principles:

- 1. Availability.** Open Standards are available for all to read and implement.
- 2. Maximize End-User Choice.** Open Standards create a fair, competitive market for implementations of the standard. They do not lock the customer in to a particular vendor or group.
- 3. No Royalty.** Open Standards are free for all to implement, with no royalty or fee. Certification of compliance by the standards organization may involve a fee.
- 4. No Discrimination.** Open Standards and the organizations that administer them do not favor one implementor over another for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low and zero-cost implementations to be validated, but may also provide enhanced certification services.
- 5. Extension or Subset.** Implementations of Open Standards may be extended, or offered in subset form. However, certification organizations may decline to certify subset implementations, and may place requirements upon extensions.
- 6. Predatory Practices.** Open Standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute, and sell software that is compatible with the extensions. An Open Standard may not otherwise prohibit extensions.

Source: © Bruce Perens <http://perens.com/OpenStandards/Definition.html>.

FSFE Open Standards Definition: An Open Standard refers to a format or protocol that is

1. subject to full public assessment and use without constraints in a manner equally available to all parties;
2. without any components or extensions that have dependencies on formats or protocols that do not meet the definition of an Open Standard themselves;
3. free from legal or technical clauses that limit its utilisation by any party or in any business model;
4. managed and further developed independently of any single vendor in a process open to the equal participation of competitors and third parties;
5. available in multiple complete implementations by competing vendors, or as a complete implementation equally available to all parties.

Source: © 2001-2011 Free Software Foundation Europe. <http://fsfe.org/projects/os/def.html>.

to use, improve upon, trust, and extend a standard over time; and freedom from all costs and tariffs associated with the above freedoms. Their definition requires, amongst others, that *patents possibly present on (parts) of the standard are made irrevocably available on a royalty-free basis*. In contrast, the Free Software Foundation Europe (FSFE) do not base their definition on cost, but more on freedom. Their definition requires that a standard is free from legal or technical clauses that limit its utilisation by any party or in any business modes. Additionally, it requires that all components or extensions that have dependencies on formats or protocols need to meet the definition of open standards.

Krechmer (1998, 2006) sets out ten principles for open standards: 1) **Open Meeting** – all may participate in the standards development process; 2) **Consensus** – all interests are discussed and agreement found, no domination; 3) **Due Process** – balloting and an appeals process may be used to find resolution; 4) **Open IPR** – how holders of IPR related to the standard make available their IPR; 5) **One World** – same standard for the same capability, world-wide; 6) **Open Change** – all changes are presented and agreed in a forum supporting the five requirements above; 7) **Open Documents** – committee drafts and completed standards documents are easily available for implementation and use; 8) **Open Interface** – supports proprietary advantage (implementation); each interface is not hidden or controlled (implementation); each interface of the implementation supports migration (use); 9) **Open Access** – objective conformance mechanisms for implementation testing and user evaluation; 10) **On-going Support** – standards are supported until user interest ceases rather than when implementer interest declines.

Krechmer also outlines the differences between his principles and the ones by Perens (no date). Krechmer indicates that Perens does not address the requirements *One World* and *On-going Support*. The principles by Krechmer are designed to address the different economic motivations of the stakeholders: while creators embrace most the principles 1-6, for developers the principles 4-9, and for users the principles 4-10 are most relevant.

7.2 Embrace, extend, and extinguish

The phrase *Embrace – Extend – and Extinguish*¹⁴ describes an internal Microsoft strategy for entering product categories involving widely used standards, extending those standards with proprietary capabilities, supporting new functionality that is taken up by the users. When these extensions become a *de facto standard*, they use the proprietary additions to the disadvantage for its competitors. This strategy has been part of a trial against the Microsoft Corporation¹⁵.

When a company uses this strategy, adding these features as FOSS could be difficult, and thus it creates disadvantages for its competitors. As a counter-measure, efforts to reverse-engineer protocols could be applied by FOSS developers. However, not in all cases this is legally or technically possible. Additionally, if the proprietary additions contain patented technologies, FOSS implementations are impossible as we will discuss in Section 7.5.

14. See http://en.wikipedia.org/wiki/Embrace,_extend_and_extinguish; accessed August 25, 2011.

15. See <http://www.justice.gov/atr/cases/f2600/2613.htm>; accessed August 28, 2011.

To avoid this, the different definitions of what an open standard is, include requirements that (a) all additions of an open standard need to be an open standard according to the same definition; or (b) the standard, including all additions need to be implementable as FOSS. The possibility to enforce such standards is that the most important and influential governments set a suitable definition of *open standards* as a requirement. In addition, they can implement these requirements for all governmental or public purchase of systems. While such a regime can force a change, we recognise that major standards follow a different policy regarding these issues.

7.3 Open Formats

An *open file format* is a published specification for storing digital data, either maintained as a standard or as a de-facto industry-defined specification that can be implemented by both proprietarily and as FOSS.¹⁶ In contrast to open formats, closed formats are considered a *trade secret*. Open formats that do not contain intellectual property rights, such as non-free licenses, patents, trademarks, or other restrictions, are denoted as *free formats*.

7.4 Standardisation and the Public Sector

Standardisation in the public sector is an important issue since a) the public sector communicates with the citizens using documents; and b) the public sector has implemented many systems that need to interact with each other¹⁷. In both cases, it is in the interest of both the public and the governmental institutions to employ as many open standards as possible¹⁸.

The public sector administrations in many countries have recognised this problem, and have imposed restrictions on which standards can be used in the public sector, i.e., require the use of open standards. However, due to the market penetration of some vendors, this cannot always be enforced.

When communicating with the citizens, there are several requirements to which document standard to use. These requirements include a) the documents need to be accessible without imposing extra licensing costs to the citizen; b) the documents need to be usable on all relevant software and hardware platforms; and c) requirements due to universal access, privacy, and other local regulations need to be satisfied. In such a definition, open standards that are implementable as FOSS will at least satisfy the requirements a) and b).

7.4.1 Document formats in the Public Sector

When communicating with the citizens, it is important that all citizens have access to the documents regardless of what software they are using. Therefore, the public sector has

16. See http://en.wikipedia.org/wiki/Open_format; accessed August 25, 2011 and www.info.org/free_file_format.html; accessed August 25, 2011.

17. In Norway, the term *samhandling* is used.

18. We recognise, that it is in the interest of some system vendors to implement proprietary technology in the public administration which can cause a user-lock-in, i.e., the user is bound to this vendor's technology base.

done efforts to standardise document formats that are open in the sense that software is available to the citizens without extra costs. In several countries the public administration has defined which document standards are allowed or preferred when communicating with the citizen.

The technologies of the W3C consortium that are used on the web, do not contain technology that cannot be implemented on all platforms, such as *HTML*¹⁹. Therefore, HTML is the preferred document format for that purpose. However, in some situations, this is not always practical when documents need to be presented in different form. Video and sound documents might also be part of the communication with the citizen, and need therefore also be openly accessible.

For read-alone text documents, also including graphics, the previously proprietary de-facto standard PDF developed by Adobe Systems²⁰ is often used. PDF was officially released as an open standard on July 1, 2008, and published by as standard ISO 32000-1:2008. Adobe also granted in a Public Patent License to ISO 32000-1 royalty-free rights for all patents owned by Adobe that are necessary to make, use, sell and distribute PDF compliant implementations.²¹

For other documents the Open Document Format for Office Applications²² (ODF), an XML-based file format for representing electronic documents such as spreadsheets, charts, presentations and word processing documents was created, originally as format for OpenOffice. ODF is an international standard: ISO/IEC 26300:2006.

In practice, in the public administration often creates documents in the Microsoft Office formats²³. However, these formats are proprietary technologies that have not been openly available^{24,25}. In order to meet the increasing requirements from many public administrations for an open standard for office applications, Microsoft has standardised the *Office Open XML* (OOXML, sometimes OpenXML)²⁶ first as ECMA-376, and later as ISO/IEC 29500.

There have been many disputes around the standardisation of OOXML and the process around it.²⁷ Note also that the OOXML specification is protected by multiple patents, where as the patent holder Microsoft corporation does not guarantee not to sue or con-

19. See <http://en.wikipedia.org/wiki/HTML>; accessed August 28, 2011.

20. See <http://en.wikipedia.org/wiki/Pdf>; accessed August 28, 2011.

21. However, making an accessible PDF document, i.e., a PDF document designed for user groups with special needs, can be difficult

22. See <http://en.wikipedia.org/wiki/OpenDocument>; accessed August 28, 2011.

23. See http://en.wikipedia.org/wiki/Microsoft_Office; accessed August 28, 2011.

24. The now published OOXML definition is an open standard. Some of the previously defined binary formats have been made available. See <http://www.microsoft.com/interop/docs/OfficeBinaryFormats.aspx>; accessed August 28, 2011.

25. Note, OpenOffice and other FOSS office systems can read and write the Microsoft Office file formats. However, there is no guarantee that the content is preserved, or that the files are compatible.

26. See http://en.wikipedia.org/wiki/Office_Open_XML; accessed August 28, 2011.

27. See <http://www.nooxml.org/>; accessed August 28, 2011, some amusing facts included: <http://www.nooxml.org/rice-pudding>; accessed August 28, 2011. Technical issues are discussed at <http://ooxmlisdefectivebydesign.blogspot.com/>; accessed August 27, 2011.

fer any other rights for competitors. OOXML also contains backwards compatibility to the older Microsoft formats, and is designed primarily for the Windows platform. Note also that the specification is not implementable a whole for competitors²⁸. Therefore, it is doubtful whether the OOXML specification qualifies as an open standard at all.

The Norwegian ministry of government administration, reform and church affairs has published a reference catalogue (Fornyings- og Administrasjonsdepartementet, 2009) for data formats to be used in the public sector. Besides HTML, PDF and ODF formats, several open multimedia formats for images, video and sound are defined as obligatory.²⁹

7.4.2 Long-term Document Storage and Digital Preservation

In the public sector, a large variety of, and a large volume of documents are produced. Many of these need to be preserved digitally, i.e., stored over a long time. Examples for such documents include publications, technical documentation, court documents, propositions, letters and minutes produced in the public sector³⁰, health care data, and tax records. Other material also include film material and other multimedia publications, books, radio transmissions that are required to be stored in the national archives by law. To store these data, it is important that open standards are used, so that these documents will remain accessible, also after the software or hardware that was used to produce these documents no longer is available.³¹

While the most obvious problems occur with documents stored on storage devices³² where the corresponding hardware no longer is available, there are also many examples of documents produced with proprietary software in proprietary data formats. Examples for these include technical drawings produced on Computer Aided Design (CAD)³³ systems and maintenance documents for buildings, ships, boats, etc. If the original system in the correct version is no longer available, e.g., the software producer has gone out of business, costly reconstruction or reverse engineering processes need to be employed to retrieve the relevant data.

To ease long-time storage, it is generally recommended to use open standards, for all files and documents, in the sense that no proprietary technology is included. Additionally, relevant procedures when handling these documents need to be implemented which assure compliance with open standards. Provided the hardware-access to the data is given, one can always implement software that provides access to these data.

For the purpose of digital preservation specific profiles of the PDF document standard

28. See <http://www.nooxml.org/argu-brief>; accessed August 28, 2011.

29. Note that in this document OOXML is defined as “under observation”.

30. For example, in Norway, all relevant documents in the public sector need to be stored according to the Offentlighetsloven.

31. There are other important issues connected with long-time storage, such as requirements to data privacy and security, as well as issues tied to DRM systems. The LongRec project, see http://www.nr.no/pages/dart/project_flyer_longrec; accessed August 28, 2010 and <http://research.dnv.com/longrec/>; accessed August 28, 2010 looked into challenges regarding long-time storage of documents.

32. See http://en.wikipedia.org/wiki/Data_storage_device; accessed August 28, 2011.

33. See <http://en.wikipedia.org/wiki/CAD>; accessed August 28, 2011.

have been standardised, denoted as *PDF/A*³⁴. The recent version PDF/A-2 (ISO 19005-2:2011) is based on the standard ISO 32000-1, but has a number of restrictions. PDF/A prohibits technologies that could cause changes with respect to the original document. Documents following this standard are not permitted to be reliant on information from external sources, such as font programs or hyperlinks. In addition, audio and video content, JavaScript, executable file launches, encryption, and certain compression methods are not permitted.

As Corrado (2005) points out, open access, open source, and open standards are important issues that can give benefits for libraries, including lower costs, better accessibility, and better prospects for long term preservation of scholarly works. Besides traditional documents, also metadata are important for digital preservation. The Open Archives Initiative³⁵ develops and promotes interoperability standards for the efficient dissemination of content. Their initiatives include both interoperability through metadata exchange and aggregation of web resources.

7.5 Patents and Standards

A *patent* is a set of exclusive rights granted by a national government to an inventor or their assignee for a limited period of time in exchange for the public disclosure of an invention.³⁶ Besides patents on inventions, there is a variety of other patents, such as *design patents* and *utility patents*³⁷ for certain types of protection rights. Previously, the term *patent* has also been used to grant certain rights to ownership and possession, and to grant the right to perform certain tasks.³⁸ In all these cases, a patent is a certificate granted by an authority that monopolises intellectual or other property rights or skills. Lately, specific types of patents for inventions have been developed, including software, chemical, medical, biological, and business method patents.

There is a conflict of interest between software patents and FOSS. According to Perens³⁹, *software patenting is generally hostile to Open Source, because patent holders require a royalty payment that isn't possible for developers who distribute their software at no charge*. Therefore, he works for reform of the patent system. Perens also reasons that *the software patenting system is broken and actually works to discourage innovation*, especially in connection to the increasingly used patenting practise of publicly funded universities. In short, patenting publicly-funded research will create injustice and economic inefficiency, since the taxpayers who indirectly funded the research might eventually get target of lawsuits. Thus, patenting works against the interest of the general public. Incorporating such patents in

34. See <http://en.wikipedia.org/wiki/PDF/A>; accessed September 1, 2011. Our thanks go to Arne-Kristian Groven who pointed out that specific standards for digital preservation have been developed.

35. See <http://www.openarchives.org>; accessed September 2, 2011.

36. See <http://en.wikipedia.org/wiki/Patent>; accessed August 28, 2011.

37. In German speaking countries the term *Gebrauchsmuster* is used as a “light-weight” patent for certain products; methods and processes cannot be protected by a *Gebrauchsmuster*.

38. In Norway, patents were given to sailors with the *sjømannspatent*, and to mountain guides with the title *patentfører*. Note that these patents both were connected to a right and a duty to perform these tasks.

39. See <http://perens.com/policy/software-patents/>; accessed August 28, 2011.

standards increases the problem, since standards are designed to agree on a common technology that is to be used by everybody, without any hindrance. Especially, this common technology should be possible to implement as FOSS.

7.6 Case Study: Video Codecs in HTML5

HTML5 (Hickson, 2011)⁴⁰ is a further development of HTML which forms the basis of today's web on the Internet. Pilgrim (2010) gives a comprehensible introduction into HTML5, and discusses its possibilities and challenges. One objective of HTML5 is to introduce support for media such as audio and video with specific tags for these. For video, the tag `<video>` has been introduced⁴¹. Besides the technical specifications, the previous draft proposal document suggested video codecs that are mandatory to be supported, while the current version is silent about this.

As previously outlined, the W3C does not allow patent-encumbered technologies to be part of their standards. Since the supported video codecs are mandatory to be implemented in all browsers without the need of plugins, the issue of patents tied to these video technologies is essential.

Multimedia content⁴² is usually delivered in a container format such as MPEG 4, Flash Video, Ogg, Audio Video Interleave (AVI), Matroska, or the newly developed WebM. These container formats contain both audio-, video- and metadata. The video-data are encoded in one of several codecs, such as MPEG-2, H.264, Theora, or VP8. Of these technologies, the H.264, Theora and VP8 are candidate technologies to be mandatory in HTML5. The HTML5 specification (Hickson, 2011) makes it clear, that the H.264 video format is not eligible to be supported mandatorily, since it is encumbered with patents.

The licensing conditions for H.264 are rather intricate, and both developers of software, as well as content distributors are subject to licensing payments administered by the MPEG LA⁴³ patents management. On the other hand, Theora and WebM are licensed royalty-free, and are not encumbered with any known patents which makes it possible to implement these codecs as FOSS. Note, however, that there always could be the risk of submarine patents⁴⁴ that could emerge in case the codec rises in popularity.

The different browsers that support HTML5 implement different selections of codecs. While browsers such as Firefox, Opera, and Chromium are in favour of Theora and WebM, others, such as Internet Explorer and Safari, choose differently⁴⁵, as do the dif-

40. The Editor's draft of this document, dated August 29, 2011 is available at <http://dev.w3.org/html5/spec/Overview.html>; accessed September 2, 2011. See also <http://en.wikipedia.org/wiki/HTML5>; accessed September 2, 2011.

41. A similar specification is used for audio with the tag `<audio>`. Unsurprisingly, for audio and other multimedia data types similar challenges as for video occur. However, for the sake of brevity we only illustrate the case for video. We refer to the book by Pilgrim (2010) for further reading.

42. We refer readers who seek deeper knowledge in multimedia formats to the advanced level course INF5081 at the University of Oslo (Leister, 2011).

43. See www.mpegla.com; accessed September 1, 2011.

44. See http://en.wikipedia.org/wiki/Submarine_patent; accessed September 1, 2011.

45. See <http://blog.chromium.org/2011/01/more-about-chrome-html-video-codec.html>; accessed

ferent mobile phones and tablets. Currently, it is not obvious how this discussion on which codecs are best supported will continue.⁴⁶ This discussion has not only an impact on openness regarding standards, but also on multimedia support for FOSS software, and on costs that arise at the content providers. Until an agreement is reached, content providers need to be prepared to store and offer video content using several types of encoding in parallel in order to reach the largest amount of users. Since also new developments, such as services for mobile devices are involved, the question of standards in multimedia formats has become a considerable factor for the further development of the information technology business.

References

- Edward M. Corrado. The importance of open access, open source and open standards for libraries. *Issues in Science & Technology Librarianship*, 42:1+, 2005. URL <http://www.istl.org/05-spring/article2.html>.
- Fornyings- og Administrasjonsdepartementet. Referanse katalog for IT-standarder Versjon 2.0. web, June 25 2009. URL http://www.regjeringen.no/upload/FAD/Vedlegg/IKT-politikk/Referanse katalogen_ versjon2.pdf. in Norwegian.
- Ian Hickson. HTML5 – a vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, May 25 2011. URL <http://www.w3.org/TR/html5/>.
- Nah Soo Hoe. *Free/Open Source Software Open Standards*. Elsevier, 2006. URL http://en.wikibooks.org/wiki/FOSS_Open_Standards.
- Ken Krechmer. The principles of open standards. *Standards Engineering*, 50(6):1–6, 1998. URL <http://www.csrstds.com/openstds.html>.
- Ken Krechmer. Open standards requirements. *International Journal of IT Standards and Standardization Research*, 4(1), 2006. URL <http://www.csrstds.com/openstds.pdf>.
- Wolfgang Leister. INF5781: Multimedia coding and applications. advanced level course, University of Oslo, 2011. URL <http://www.uio.no/studier/emner/matnat/ifi/INF5081/>.
- Bruce Perens. Open standards, principles and practice. web pages, no date. URL <http://perens.com/OpenStandards/Definition.html>. last modified October 13, 2010.
- Mark Pilgrim. *HTML5: Up and Running*. Google Press, 1 edition, August 2010. ISBN 0596806027. URL <http://fortuito.us/diveintohtml5/>. available as ebook under the title “Dive into HTML5” on mirror site, CC-BY-3.0 License.

September 1, 2011.

46. See <http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2009-June/020620.html>; accessed September 1, 2011.