Chapter 4

Open Source Software (OSS)

> *"Gnu: n. [Hottentot gnu, or nju: cf. F. gnou.] (Zo["o]l.) One of two species of large South African antelopes of the genus Catoblephas, having a mane and bushy tail, and curved horns in both sexes. [Written also gnoo.]*
>
> *Note: The common gnu or wildebeest (Catoblephas gnu) is plain brown; the brindled gnu or blue wildebeest (C. gorgon) is larger, with transverse stripes of black on the neck and shoulders." — Webster's Revised Unabridged Dictionary.*

Open Source Software (OSS) [92], also known as free software [105], is any software distributed under a license conforming to the Open Source Definition (OSD) as published by the Open Source Initiative (OSI)[1]. An unannotated copy of the current OSD is attached as Appendix C, however, later versions may be published on the OSI web site as the definition is refined. An annotated version, describing the motivation for each clause of the definition, is also available from the OSI web site. Unlike the OSI, which approaches OSS from a very pragmatic perspective, the Free Software Foundation (FSF)[2] approaches OSS from a more ethical ideology concerned with civil liberties. Essentially, free software licenses are designed to protect four basic freedoms:

- **Freedom of use:** Recipients of OSS are granted the right to use the software for any purpose.

[1] http://www.opensource.org
[2] http://www.fsf.org

- **Freedom to source:** Recipients of OSS are provided free access to the source code.
- **Freedom to modify:** Recipients of OSS are granted rights to prepare derivative works.
- **Freedom to distribute:** Recipients of OSS are granted rights to distribute the software, in original or modified form, either for free or for a fee.

While the OSI and FSF have somewhat different motives and are in disagreement about whether OSS should properly be called free software and *vice versa*, a common ground lies in the terms of the licenses that they both advocate. Therefore, the most popular OSS licenses and their characteristics are surveyed in Section 4.1.

OSS has many benefits for both developers and users of the software. From the user perspective, the zero marginal cost and high quality of OSS are often cited. Section 4.2 discusses the OSS ecosystem while concentrating on the benefits of OSS to developers. A common misconception regarding OSS is that it cannot be utilised for financial gain, however, it is certainly possible to make money from OSS through indirect sale business models such as those mentioned in Section 4.3. In fact, many large industry players such as IBM[3], Sun Microsystems[4] and Novell[5] have embraced OSS for profit.

OSS is of particular importance to developing countries. In particular, Section 4.4 discusses OSS in a South African context. Further, certain software pertaining to this work is distributed under an OSS license. Since this work constitutes University of Pretoria intellectual property, strong motives for releasing the software under such a license are provided in Section 4.5. Finally, this chapter concludes with credits in Section 4.6, listing the OSS that has been instrumental in completing this work.

4.1 Licenses

The characteristics of the most popular[6] and best known OSS licenses are compared in Table 4.1. The complete text of these licenses are provided in Appendix E as a reference.

[3] http://www.ibm.com
[4] http://www.sun.com
[5] http://www.novell.com
[6] According to SourceForge, http://sourceforge.net/softwaremap/trove_list.php?form_cat=14

Terms and conditions for many other free software licenses are available on the OSI and FSF web sites. In addition, many OSS licenses have multiple versions and it should be noted that this work only considers the latest versions of those licenses at the time of writing. Newer versions will more than likely be published by the OSI or FSF as they come to exist.

While all of the licenses listed in Table 4.1 are OSI approved and are classified as free software licenses in terms of the four freedoms presented at the beginning of this chapter, they can be further divided into two broad categories: those that are copyleft, or GPL style; and those that are not, such as the BSD or MIT style licenses. Copyleft licenses place an additional restriction on the software, so they are less permissive and are therefore arguably less free licenses, requiring that any modifications, if distributed, must be made available under free terms again. A copyleft clause in a license essentially prevents free software from becoming non-free, which benefits the free software community as a whole even though the rights of any given individual within that community are curtailed.

The GNU General Public Licence (GPL), developed by the FSF as the license for the GNU Project[7], is probably the most important free software license in existence, with in excess of 39 thousand SourceForge[8] software projects licensed under its terms, including software developed for this work. The compatibility of other licenses to the GPL is an important characteristic of a license, since software licensed under incompatible terms cannot be linked against GPL software.

Table 4.1 further characterises licenses based on whether they permit additional warranty or liability protection to be sold and whether the license grants patent rights in addition to the four basic freedoms of free software.

Sections 4.1.1 through 4.1.9, in turn, detail the characteristics of each of the licenses presented in Table 4.1.

4.1.1 Academic Free License (AFL)

The Academic Free License (AFL, version 2.1), in Appendix E.1, is a non-copyleft license provided by the OSI. Software specific details are avoided in the license terminology,

[7]GNU: A recursive acronym for GNU's Not Unix; refer to http://www.fsf.org/gnu/thegnuproject.html for information about the GNU Project

[8]http://www.sourceforge.net

Table 4.1: Open Source License Characteristics

License	Copyleft	GPL Compat.	Warranty Prov.	Patent Lic.
AFL	X	X	X	√
ASL	X	X	√	√
AL	X	√	√	X
BSD (original)	X	X	X	X
BSD (revised)	X	√	X	X
CPL	√	X	√	√
GNU GPL	√	-	√	X
GNU LGPL	-	√	√	X
MIT	X	√	X	X
MPL	-	-	√	√
OSL	√	X	X	√

making the license ideally suited for non-software works, such as documents, while still being general enough to apply to software.

The second clause grants a recipient of a work covered by the license a royalty-free right to use and sub-license patents. In addition, if a recipient enters into any patent infringement action against a licensor or licensee, that recipient's rights under the license are terminated. The patent termination clause makes the AFL incompatible with the GPL.

No provision is made for a licensor to sell additional warranty or liability protection. The work is licensed as is, without any warranties, aside from a warranty that applicable copyrights and patents are owned by the licensor, and disclaims all liability.

4.1.2 Apache Software License (ASL)

The Apache Software License (ASL, version 2.0) is a free software license with similar patent grant and termination clauses to the AFL, also making it incompatible with the GPL. Clause 9 permits anyone who distributes software under the ASL to provide additional warranty or liability protection. Finally, the license is not copyleft, meaning that any recipient may distribute the software under different license terms as long as

all the obligations of the ASL, as specified in Appendix E.2, are met.

4.1.3 Artistic License (AL)

The Artistic License (AL, version 2.0beta4), presented in Appendix E.3, is designed to protect an originator's artistic control over future versions of the software. In essence it requires modified versions to clearly indicate any changes and satisfy one of the following conditions: i) the changes must be submitted back to the original contributor for consideration in the standard version, ii) the package must be renamed to something that cannot be confused with the original, or iii) it must be made available under free terms to whomever it is distributed to.

Although the AL is scattered with hints of copyleft concepts, clause 6(b) clearly allows the software to be made non-free, so long as any changes are documented and that it cannot be confused with the original work. The license is, however, GPL compatible and although no specific clause specifically applies to additional warranty provisions, the standard warranty disclaimer text, in clause 11, does permit such provisions to be stipulated in writing. Patent licenses are not covered.

4.1.4 BSD Licenses

The revised BSD license, presented in Appendix E.4, is an extremely permissive non-copyleft license which primarily ensures that copyright notices are properly maintained. The original version had an additional advertising clause, requiring the University of California, Berkeley and its contributors to be credited in any advertising material, making it incompatible with the GPL. Neither version permits the names of any contributors to be used as an endorsement to promote the licensed work. Both forms of the license explicitly disclaim all liability and warranties while saying nothing about patents.

4.1.5 Common Public License (CPL)

The Common Public License (CPL, version 1.0), in Appendix E.5, has been designed to facilitate the commercial use and distribution of software. The CPL is not compatible with the GPL. It has similar patent grant and termination clauses to the AFL and ASL, but unlike those licenses, it offers some copyleft characteristics.

The copyleft terms in the CPL are not as stringent as the GPL, since separate modules may be licensed under their own terms. While derivative works are explicitly excluded from this concession, it is not explicitly clear where the boundary between a module and a derivative work lies. Binary distribution under another license is also permitted provided that i) warranty and liability exclusions are carried over, ii) source code is made available to a recipient on request, and iii) the terms of the other license do not otherwise conflict with requirements of the CPL.

Warranty and other liability protections may be offered provided that any other contributors are properly indemnified. That is, a distributor offering additional protections accepts all responsibility, including defending any legal claims made against any contributor.

4.1.6 GNU General Public Licenses (GPL and LGPL)

The GNU General Public License (GPL, version 2), presented in Appendix E.6, is a strong copyleft license. In fact, the GPL is the original definition of copyleft. That is, the copyleft terminology was coined by the FSF to encompass those properties of the GPL that keep software free. In the case of the GPL, copyleft is accomplished by requiring that any derivative work must again be distributed under the free terms of the GPL, if it is distributed at all. As a consequence, if a portion of a work is licensed under the GPL then the whole may not be distributed at all, except under terms of the GPL, since the whole would qualify as a derivative work.

On the other hand, the GNU Lesser, or originally Library, Public License (LGPL, version 2.1), in Appendix E.7, has more relaxed copyleft requirements. The LGPL was originally written to enable a free software library to be used by a non-free, or proprietary, work without requiring the whole to be made freely available. However, any improvements or other changes to the library itself are still required to be distributed under the free terms of the LGPL. That is, a work covered by the LGPL will remain free while any other separate work that links against it, technically a derivative work, is not required to be released under the terms of the LGPL. Since the LGPL is applicable to more works than just libraries, it was renamed the Lesser GPL, to reflect the less stringent copyleft requirements. Any recipient of a LGPL work may choose to redistribute it under the more restrictive copyleft terms of the GPL.

Incompatibilities with the GPL arise from clauses 6 and 7 of the GPL, which state that a distributor may not impose any further restrictions on a recipient beyond what the terms of the GPL permit. To do so would render a work undistributable under the GPL. For example, a condition of the GPL is a royalty free right to use the software licensed under its terms, however, if a combined work consists of some non-GPL portions which would prevent such royalty free use, perhaps due to a patent, then the right to distribute the GPL portion falls away too, leaving the whole in a state which cannot be distributed under either license. For this reason, patent termination clauses in other licenses cause an incompatibility with the GPL. Neither the GPL nor the LGPL explicitly include a patent grant, however, clauses 6 and 7 do provide free software with a certain level of protection from patents, in so far as the free software cannot be distributed by a patent holder under terms other than the GPL. The FSF has recently announced plans to release a new version of the GPL[9], which is likely to have patent terms that are more compatible with other popular OSS licenses. Since the LGPL is essentially the same as the GPL, except for the more lenient copyleft terms, it is GPL compatible.

Both the GPL and LGPL grant distributors of software the freedom to offer additional warranties or liability cover to their recipients.

4.1.7 MIT License

The MIT license, presented in Appendix E.8, is probably the least restrictive free software license. Permission to use, modify and distribute the software is granted provided that the copyright and permission notice is preserved. The permission notice also includes a simple disclaimer which explicitly disclaims any liability or warranties. Since it essentially does not place any restrictions on the software it covers, it is GPL compatible and non-copyleft.

4.1.8 Mozilla Public License (MPL)

The Mozilla Public License (MPL, version 1.1), in Appendix E.9, has similar copyleft properties to the LGPL. Clause 3.7 permits a larger work to be composed and distributed under a different license provided that the MPL requirements are fulfilled for the covered code. In addition, patent licenses, subject to litigation termination terms, are granted

[9] http://www.eweek.com/article2/0,,1730102,00.asp

by the MPL. Clause 3.5 explicitly provides for warranty support or liability obligations under the condition that other contributors are properly indemnified. Finally, an initial developer may, subject to clause 13, choose to license portions, or the whole, of the work under multiple license terms, including GPL, making those parts GPL compatible.

4.1.9 Open Software License (OSL)

The Open Software License (OSL, version 2.1), presented in Appendix E.10, is virtually identical to the AFL, except that copyleft properties are ensured by clause 1c, which requires derivative works to be distributed under terms of the OSL. Like the AFL, the OSL grants patent licenses, is not GPL compatible and makes no provision for additional liability or warranty cover.

4.2 The Open Source Ecosystem

Hardin's tragedy of the commons describes the inevitable demise of any freely shared resource, the commons, if no resource allocation policy is enforced [51]. As an example, Hardin considers the scenario of a public pasture which is freely shared amongst a number of cattle farmers. The grazing cost, in terms of damage to the pasture, of another head of cattle is diluted by the commons, while any given farmer still retains the full profits associated with owning more cattle. This imbalance gives each farmer the incentive to add more and more cattle, to extract the maximum value from the commons as quickly as possible before it degrades due to over grazing. There is no incentive to contribute to the maintenance of the commons, since any returns would again be diluted.

Freely available OSS, however, does not suffer this tragedy [92, 44, 104]. There are two contributing factors to the tragedy of the commons: i) there is a limited supply of resources; and ii) the lack of an enforced allocation policy drives demand up until the supply is depleted. Fortunately, in today's Internet connected world, software costs virtually nothing to duplicate. As a resource, software is not depleted by the act of copying, so free riders do not degrade the commons. On the contrary, a larger user base actually increases the value of OSS. Thus, the demand side of the equation is taken care of, and tragedy is avoided. On the other side of the equation, there are strong incentives for developers to contribute to the commons, ensuring sufficient supply of free software.

Compelling reasons why people and organisations contribute to the free software commons include:

- **Peer review and reputation rewards:** A large user base can be a free software project's biggest asset. Aside from the benefits of having users provide bug reports and feature requests, high profile projects also offer the highest reputation rewards, attracting the attention and cooperation of other developers. The peer review process associated with more developers, in turn, improves the quality of the software.

- **Cost and risk sharing:** Customising existing free software to meet the specific needs of a user can be cheaper than developing a solution from scratch. Further, there is a strong incentive to contribute any improvements back to the community, even ignoring possible copyleft constraints on the existing software. To see why, consider the situation where a user chooses not to contribute those improvements back to the community. Now, that user needs to maintain a separate version, possibly merging it with improvements from the community version from time to time. This can be an expensive undertaking, particularly if the community version undergoes incompatible changes. Contributing the changes back avoids this problem. Thus, it is a reasonable assumption for an initial contributor of software to expect others to contribute improvements, initiating a cost sharing development excersise. Also, the community offers safety. The risk of having only a few people being able to maintain the software can be mitigated by sharing that maintenance burden with the community, so that more than one entity has a vested interest in the survivability of the software.

- **Growing secondary markets:** Very importantly, there is money to be made from free software. By growing the community around a free software product, related secondary markets are opened up. The indirect sale business models presented in the next section exploit this property.

4.3 Business Models

OSS licenses typically do not prevent the distribution of software for a fee, however, some do require that such a fee be at most the reasonable cost of copying. More importantly,

all OSS licenses explicitly grant any recipient the right to freely distribute the software again, making it difficult to build a direct sale business based on OSS. That said, several indirect sale business models exist to exploit free software for financial gain [92]:

- **Loss leader/market positioner:** Free software is used to maintain or create a secondary market for other non-free software. Thus, the use of the free product drives sales of the non-free product. For example, giving away free development tools in order to maintain the market for application servers, which is what IBM is doing with the Eclipse platform to drive sales of their WebSphere[10] solution.

- **Widget frosting:** Hardware products typically require accessory software which does not have any sale value independent of the hardware. For example, drivers or configuration software. By opening up the software, a hardware vendor can benefit from a larger developer pool, better reliability through peer review and maintenance beyond the expected product life cycle. All without sacrificing any revenue stream, since it is the hardware that brings in the money. A concrete example is Apple's[11] decision to open source Darwin, the core of MacOS X, since they are primarily interested in selling the hardware on which the operating system runs.

- **Give away the recipe, open a restuarant:** The software is provided freely and services are sold to the created market. For example, vendors may choose to sell support contracts, performance assurances, customisation services, training and maintenance of the software according to the client's time table. RedHat[12], for example, sells support and patch management for their open source Linux product.

- **Accessorising:** Accessories to the software are sold. Trivial examples include mugs and t-shirts, while publishers such as O'Reilly sell high quality books about free software. Other accessories might include non-free plug-ins that enhance the functioning of the software.

- **Free the future, sell the present:** Software is initially sold under a closed license with the provision that it will be released under a open license at a later

[10] http://www-306.ibm.com/software/webservers/appserv/was/
[11] http://www.apple.com
[12] http://www.redhat.com

date. Sales volumes are driven by the expectation that the software will become free later, while the vendor benefits from the reduced maintenance overhead later in the product life cycle.

- **Free the software, sell the brand:** The software implementation is free. Customers must satisfy compatibility requirements and pay for the certification of the brand.
- **Free the software, sell the content:** The software is free, while content subsciptions are sold. For example, a game engine might be given away freely while the story is sold for a price.
- **Dual licensing:** This model requires the vendor to own, or at least control, all copyrights pertaining to the software. The product is released to the public under a strong copyleft license, such as the GPL, making it impossible to distribute the free software component as part of other non-free commercial software. Simultaneously, the software is sold, under a non-free license, to clients that wish to incorporate the software into commercial software. A community is built around the free version of the software, building market awareness of the product. Typically, improvements from the community may only be incorporated into the non-free version with the permission of a contributor. Vendors may require copyrights to be signed over in order for improvements to be incorporated into the free reference version. Dual licensing has been successfully employed by MySQL[13], for their database product, and Sun Microsystems[14], for their StarOffice product which is available in a scaled down form as OpenOffice[15].

The common theme amongst open source business models: software is provided for free to produce a secondary market where additional value can be sold for a price.

4.4 Open Source in a South African Context

An official open source strategy [3] has been proposed by the local South African government. The proposal addresses the benefits of OSS in a South African context, rec-

[13] http://www.mysql.com
[14] http://www.sun.com
[15] http://www.openoffice.org

ommendations for building local competencies in open source and a long term strategy for providing government support for open source projects.

Key economic benefits, amongst others identified in the report, are the development of local software development skills and the saving of foreign currency, since most commercial software is developed abroad. By leveraging open source as an educational vehicle, local skills in software development are developed, which in turn will stimulate SMME (Small, Medium and Micro Enterprises) growth in the IT (Information Technology) sector. Some responsibilities (quoted directly from the report) of educational institutions for building a capacity in open source are:

- "It is critical that strong linkages be set up with institutions of higher learning to build a national collaborative network that can be extended internationally."
- "Training for OSS developers and OSS users must be available. Institutions of learning must fulfil a role in this respect."
- "A well-run research programme will be needed to enable optimal understanding and decision making on OSS. The model for this research programme should be built on the networking nature of the OSS development model, harnessing the potential of institutions of higher leaning and schools."

The advantages that OSS holds for the local economy makes it the responsibility of every South African citizen to leverage OSS whenever it makes business sense, reducing foreign spending on software and creating a demand for local skills in the secondary markets discussed in the previous section. The Shuttleworth Foundation[16] is setting a fine example by actively promoting OSS in South Africa, targeting the general public with a wide reaching "Go Open Source"[17] awareness campaign, and facilitating the use of OSS in schools.

[16] http://www.shuttleworthfoundation.org
[17] http://www.go-opensource.org/

4.5 University of Pretoria Intellectual Property

The University of Pretoria (UP)[18], like most universities, retains ownership of any Intellectual Property (IP) submitted by students for degree purposes[19]. This means that any decision to license the source code pertaining to this work, which is material covered by UP copyrights, to third parties legally rests with the university's IP authorities. Therefore, permission to publish the CILib source code under the GPL needed to be granted officially. A draft of the letter granting this permission is included as Appendix D. The following reasons were offered as motivation for obtaining this permission:

- **Collaboration, reputation and peer review:** The CIRG@UP would like to solicit the collaboration of third parties to accelerate the development of CILib through a mutually beneficial sharing of development resources. By releasing the source code under the GPL, the group hopes to benefit from the OSS peer review process, with a goal of producing a reliable and error free software platform capable of engendering a community's trust in its code base. Further, the copyleft nature of the GPL should encourage those who find the software useful to contribute any improvements they may make back to the community. If successful, the University of Pretoria, as initial contributor and founder of the community, will benefit from the reputation associated with such a project.

- **Use of other GPL software:** Distributing software under the GPL enables it to incorporate other GPL software. For example, CILib makes use of simulation quality random number generators ported from the GNU Scientific Library (GSL)[20], which is only licensed to the university under the GPL. This also means that CILib may not be distributed under any license terms besides the GPL. At that time, the university could have chosen not to distribute the software at all, keeping it secret and losing out on all the other benefits mentioned here. Since the university currently owns the rest of the copyrights pertaining to CILib, it may choose to distribute those components which it owns under its own terms at any point in the future. That is, provided the GSL components are removed, that version of CILib may be licensed under other terms, however, the quality of any simulations

[18] http://www.up.ac.za
[19] According to the contract signed by students upon application for a degree.
[20] http://www.gnu.org/software/gsl/

performed using the software would be severely diminished, reducing the value of the software as a product. Note that nothing can retroactively revoke any rights that the university has granted to any third party who has already received a copy of CILib under the GPL.

- **Social Responsibility in a South African context:** Given the discussion in the previous section, it is important for the university to be a good citizen of the open source community. In fact, the UP is actively pursuing open source research through initiatives such as digital@SERA [111], a division of the Southern Education and Research Alliance (SERA)[21] which is a joint venture between the UP and the CSIR (Council for Scientific and Industrial Research)[22] focused on fostering collaborative and sustainable research. CILib is simply another opportunity to develop local skills while researching the applicability of the OSS development model with respect to collaborative research.

- **Business opportunities:** Building a community around a freely available software product creates the potential to exploit secondary markets, due to increased visibility of the product in the market place.

  In the case of CILib (refer to Chapter 6), it is conceivable that a future third party might like to utilise the software in a commercial product offering. As discussed previously, the university may license the software on its own terms to such a third party for a fee, provided it satisfies its GPL obligations, by excluding any GPL material not covered by university copyrights. Further, the university may be able to co-operate in some kind of profit sharing scheme with other copyright holders to offer a product of increased value to commercial third parties. Policies requiring potential contributors to sign over their copyrights or grant permission for their work to be included commercial offerings should be avoided, since such policies may discourage contributions.

  For CiClops (refer to Chapter 7), the CIRG@UP is still undecided as to an appropriate course of action. The university may choose to keep it proprietary, following a "loss leader/market positioner" business model. Under this model, CiClops is used to maintain a central repository of CILib simulation data while selling the

[21] http://www.seralliance.com/
[22] http://www.csir.co.za

services of the software and the use of the data repository. The difficulty will be gaining the trust of third parties, if they cannot access the source code, they cannot verify the correctness of the software or the integrity of the data repository. On the other hand, a "free the software, sell the content" model which does not have this problem could be pursued. In this model, only the data repository and university computing resources are sold as a service. The danger with this is that it opens the door to competing repositories, discouraging collaboration on a single data repository.

4.6 Credits

Table 4.2: Instrumental Open Source Software

Package	**License**	**Web Site**
Apache Ant	ASL	http://ant.apache.org
CVS	GNU GPL	http://www.cvshome.org
Dia	GNU GPL	http://www.gnome.org/projects/dia/
Eclipse	EPL	http://www.eclipse.org
Emacs	GNU GPL	http://www.gnu.org/software/emacs/emacs.html
GNU/Linux	GNU GPL	http://www.fsf.org/gnu/linux-and-gnu.html
		http://www.gentoo.org
JBoss	GNU LGPL	http://www.jboss.org
JUnit	CPL	http://www.junit.org
Mozilla	MPL	http://www.mozilla.org
MySQL	GNU GPL	http://www.mysql.com
NetBeans	SPL	http://www.netbeans.org
teTeX	Various OSS	http://www.tug.org/teTeX/
XDoclet	BSD (revised)	http://xdoclet.sourceforge.net/xdoclet/index.html
Xfig	Xfig custom	http://www.xfig.org

Table 4.2 presents a list of free software titles which can be credited for making this work possible. The distribution license and web site where futher information can be obtained

are also listed alongside each title.

On the software implementation front, Eclipse, distributed under the Eclipse Public License (EPL), and NetBeans, distributed under the Sun Public License (SPL), have both been used as development environments. Software version control is maintained using the CVS (Concurrent Versioning System), since it is the only version control system currently supported by SourceForge. A recent SourceForge circular announced plans to support the more modern Subversion[23] system in the near future. The Apache project's Ant is the tool used to script the build process for all the software developed for this work. Software unit testing is performed using the JUnit framework. Components of the software are deployed on a JBoss application server using XDoclet to generate the necessary deployment descriptors and ancillary interfaces. MySQL has been used to provide the relational database back-end used by the application server.

This dissertation has been composed using the Emacs text editor and typeset with the teTeX LaTeX processor. All UML diagrams were composed using Dia, while the remaining figures have been drawn using Xfig. The Mozilla browser has been used for researching resources on the web. Finally, underlying all this excellent software has been the GNU/Linux operating system. This work would not have been possible, at least not within budget constraints, without the aid of free software.

[23] http://subversion.tigris.org/