

Copyright © 2006

Pakistan Software Export Board (G) Limited
Ministry of Information Technology
Government of Pakistan

Printing

Artland Communications, Lahore. September 2006

Published by

Pakistan Software Export Board

The Funding Agency

This open source toolkit is funded by the Open Source Resource Center (OSRC) project of the Pakistan Software Export Board (PSEB). PSEB is the entity within Government charged with the task of enhancing exports of software and IT enabled services (ITES) from Pakistan. PSEB is a guarantee limited company totally owned and funded by the Government of Pakistan. Any questions or comments about this toolkit may be directed to PSEB Islamabad at 92-51-111-333-666 or through e-mail at osrc@pseb.org.pk.

Disclaimer

This toolkit is published by the PSEB for members of the IT industry and the public-at-large. The toolkit's compilers, or the editor, are not responsible, in any way possible, for the errors/omissions of this toolkit. The OSRC does not accept any liability for any direct and consequential use of this toolkit or its contents. The contents of this toolkit may be distributed only subject to the terms and conditions set forth in the Open Publication License v 1.0 or later. The latest version is presently available at <http://opencontent.org/openpub/>

TABLE OF CONTENTS

INTRODUCTION.....	1
HISTORY AND LICENSING OF FOSS.....	2
1. WHAT IS FREE/OPEN SOURCE SOFTWARE (FOSS)?.....	3
2. THE FOSS PHILOSOPHY.....	3
3. THE FOSS DEVELOPMENT METHOD.....	3
4. THE HISTORY OF FOSS.....	5
4.1. A Brief History of Free/Open Source Software Movement.....	5
5. WHY FOSS?.....	6
6. THE BENEFITS OF USING FOSS.....	6
6.1. Security.....	6
6.2. Reliability/Stability.....	7
6.3. Open standards and vendor independence.....	7
6.4. Reduced reliance on imports.....	8
6.5. Developing local software capacity.....	8
6.6. Piracy, Intellectual Property Rights (IPR), and the World Trade Organization (WTO).....	9
6.7. Localization.....	9
7. THE SHORTCOMINGS OF FOSS.....	9
7.1. Lack of business applications.....	9
7.2. Interoperability with proprietary systems.....	9
7.3. Documentation and “polish”.....	10
8. THE LICENSING ARRANGEMENTS FOR FOSS - AN OVERVIEW OF INTELLECTUAL PROPERTY RIGHTS.....	10
8.1. How is software protected?.....	10
8.2. Copyright Basics.....	11
8.2.1. What can be copyrighted?.....	11
8.2.2. Is anything required to be done in order to get a work protected?.....	11
8.2.3. What are the rights granted to the copyright-holder?.....	11
Copyright law protection has been expanded largely with time.....	11
8.3. The Expansion of Copyright Protection.....	11
8.3.1. The first copyright legislation (Statute of Anne, 1710).....	11
8.3.2. All-dimensional expansion of copyright protection.....	11
8.4. From national to international protection.....	12
8.4.1. Berne Convention.....	12
8.4.2. A more enforceable international standard- WTO and TRIPs.....	12
8.4.3. Protection upon creation, the abolishment of formality requirements.....	12
8.4.4. Copyright law, a balance between public and private interests.....	12
8.5. Software and Copyright Protection.....	13
8.5.1. The extension of copyright law protection to software in the 1980s.....	13
8.5.2. Copyright protects both the source and the object code under TRIPs.....	13
8.5.3. Users’ rights disguised in proprietary licensing modds.....	13
9. HOW IS FOSS DIFFERENT FROM PROPRIETARY SOFTWARE?.....	14
9.1. Free Software.....	14
9.2. Richard Stallman on a stark moral decision.....	14
9.3. Free Software Definition.....	14
10. CREATING A FREE SOFTWARE ENVIRONMENT.....	15
10.1. The GNU Project and the Free Software Foundation.....	15
10.2. GNU General Public License (GNU GPL or GPL).....	15
11. OPEN SOURCE SOFTWARE.....	15
11.1. Open Source Definition.....	16
12. OSI-APPROVED LICENSES.....	16
12.1. Free or Restrictive?.....	16
12.2. How to make the source free/open?.....	17
12.3. A Comparison of FOSS Licenses v2.1.....	18
12.4. GNU General Public License (GNU GPL or GPL).....	20
12.4.1. Copyleft.....	21
12.4.2. Major terms and conditions of the GPL.....	21
12.5. GNU Lesser General Public License (GNU LGPL or LGPL).....	22
12.5.1. Major terms and conditions of the LGPL.....	22
12.6. Berkeley Software Distribution (BSD)-style Licenses.....	23
12.7. Multiple Licensing.....	23

<u>12.8. The source code is open: what about documentation?</u>	<u>24</u>
<u>12.8.1. GNU Free Documentation License (GNU FDL or FDL)</u>	<u>24</u>
<u>12.8.2. Creative Commons Licenses</u>	<u>24</u>
<u>13. SCENARIOS</u>	<u>25</u>
<u>13.1. End-user (individual/business/government)</u>	<u>25</u>
<u>13.1.1. Legal Issues Involved</u>	<u>25</u>
<u>13.2. Developer (individual, business)</u>	<u>26</u>
<u>13.2.1. When starting a new project</u>	<u>26</u>
<u>13.2.2. When modifying an existing module</u>	<u>27</u>
<u>13.2.3. When integrating different FOSS modules into one service</u>	<u>28</u>
<u>13.3. Vendor/Producer (Business)</u>	<u>29</u>
<u>13.3.1. Simple Distribution</u>	<u>29</u>
<u>13.3.2. Distribution of Integrated Systems</u>	<u>29</u>
<u>13.3.3. Government-sponsored Projects</u>	<u>29</u>
<u>14. GLOSSARY</u>	<u>32</u>

Introduction

This open source toolkit has been developed by the Open Source Resource Center (OSRC), a project of the Ministry of Information Technology (MoIT). This toolkit contains step-by-step manuals related to open source applications for databases, application servers, desktop applications, office productivity suites, Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) software, and open source desktop applications for the Microsoft Windows platform. A set of CDs, including some Linux distributions and other applications, forms an integral part of this open source toolkit.

I would like to thank the OSRC team, including Mr. Abubakar Shoaib, Mr. Iftikhar Ahmad, Mr. Muhammad Hammmad, Mr. Muazzam Ali, Mr. Sher Shah Farooq, and Mr. Qandeel Aslam, who have compiled this toolkit; and Miss Seema Javed Amin, who has edited it. The OSRC would especially wish to thank PSEB's Director (Projects) Mr. Nasir Khan Afridi, Former Project Manger(OSRC) Mr. Osman Haq and Ministry of Information Technology's Member (IT) Mr. M. Tariq Badsha for their generous moral support, without which this toolkit would never have been completed.

This is the first edition of this toolkit, and the OSRC hopes to continue to improve it with the help of your feedback and comments.

Sufyan Kakakhel

Open Source Resource Center,
Pakistan Software Export Board,
2nd Floor, ETC, Agha Khan Road, F-5,
Islamabad, Pakistan.
Ph: +92-51-9208748
Fax: +92-51-9204075
Email: skakakhel@pseb.org.pk
<http://www.osrc.org.pk>

History and Licensing of FOSS

1. What is Free/Open Source Software (FOSS)?

“Briefly, OSS/FS programs are programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program (without having to pay royalties to previous developers).”

David Wheelerⁱ

Free and Open Source Software (FOSS) has become an international phenomenon, moving from relative obscurity to being the latest buzzword in a few short years. There is, however, still a lack of understanding about what really constitutes FOSS and the ramifications regarding this new concept. To better explain this phenomenon, the philosophy and development methods behind FOSS are given below.

2. The FOSS Philosophy

There are two major philosophies in the FOSS world: the Free Software Foundation (FSF) philosophy and the Open Source Initiative (OSI) philosophy. We begin with the FSF philosophy due to its historical precedence (see the following section, “A Brief History of FOSS”) and its pioneering position in the movement.

According to the FSF, free software is about protecting four user freedoms:

- The freedom to run a program, for any purpose
- The freedom to study how a program works and adapt it to a person’s needs. Access to the source code is a pre-condition for this
- The freedom to redistribute copies so that you can help your neighbour
- The freedom to improve a program and release your improvements to the public, so that the whole community benefits. Access to the source code is a pre-condition for thisⁱⁱ

At the heart of the FSF is the freedom to cooperate. Because non-free (free as in freedom, not price) software restricts the freedom to cooperate, the FSF considers non-free software unethical. The FSF is also opposed to software patents and additional restrictions to existing copyright laws. All of these restrict the four user freedoms listed above. For a more detailed explanation of why software needs to be free, please refer to the FSF explanation, “Why Software Should Be Free”, found at <http://www.fsf.org/philosophy/shouldbefree.html>

The OSI philosophy is somewhat different:

The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.ⁱⁱⁱ

The OSI is focused on the technical values of making powerful, reliable software, and is more business-friendly than the FSF. It is less focused on the moral issues regarding free software, and more focused regarding the practical advantages of the FOSS distributed development method.

While the fundamental philosophies of the two movements are different, both the FSF and the OSI share the same space and cooperate on practical grounds such as software development, efforts against proprietary software, software patents, and the like. As Richard Stallman says, the free software movement and the open source movement are two political parties in the same community.

3. The FOSS Development Method

The FOSS development model is unique and became possible only with the advent of the Internet and the communication boom caused by it. “The Cathedral and the Bazaar” analogies^{iv} are used to contrast the FOSS development model with more traditional software development methods.

Traditional software development is likened to the way cathedrals were built in ancient times. Small groups of skilled artisans carefully planned out the design in isolation and everything was built in a single effort. Once built, the cathedrals were complete and little further modification was made. Software was traditionally built in a similar fashion. Groups of programmers worked in isolation, with careful planning and management, until their work was completed and the program released to the world. Once released, the program was considered finished and, subsequently, limited work was done on it.

In contrast, FOSS development is more akin to a bazaar, which grows organically. Initial traders come, establish their structures, and begin business. Later traders come and establish their own structures, and the bazaar grows in what appears to be a very chaotic fashion. Traders are concerned primarily with building a minimally functional structure so that they can begin trading. Later additions are added as circumstances dictate. Likewise, FOSS development starts off highly unstructured. Developers release early minimally functional code to the general public and then modify their programs based on feedback. Other developers may come along and modify or build upon the existing code. Over a period of time, an entire operating system and suite of applications develops and evolves continuously.

The “bazaar” method of development has been proven over time to have several advantages:

Reduced duplication of effort: By releasing programs early and granting users the right to modify, and redistribute, the source code, FOSS developers reuse the work produced by compatriots. The economies of scale can be enormous. Instead of five software developers in ten companies writing a single networking application, there is the potential for the combined efforts of fifty developers. The reduced duplication of effort allows FOSS development to scale to massive, unheard of levels involving thousands of developers around the world.

Building upon the work of others: With the availability of existing source code to build on, development times are reduced. Many FOSS projects rely on software built by other projects to supply needed functionality. Instead of writing their own cryptographic code, for example, the Apache web server project uses the OpenSSL project’s implementation, thereby saving thousands of hours of coding and testing. Even in cases where source code cannot be directly integrated, the availability of the existing source code allows developers to learn how another project has solved a similar problem.

Better quality control: “Given enough eyeballs, all bugs are shallow”^v is an oft-cited quotation in the FOSS world. It means that, with enough qualified developers using the application and examining the source code, errors are spotted and fixed at a much faster rate. Proprietary applications may accept error reports, but because you, the user are denied access to the source code, you are strictly limited to merely reporting symptoms. FOSS developers often find that users with access to the source code not only report problems, but also pinpoint the exact cause and, in some cases, supply the fixes as well. This greatly reduces development and quality control time.

Reduced maintenance costs: Maintenance of any software package can often equal, or exceed, the cost of initial software development^{vi}. When a single organization has to maintain software, this can be an extremely expensive task. With the FOSS development model, however, maintenance costs can be shared among the thousands of potential users of a software application, reducing per-organization costs. Likewise, enhancements can be made by the organization and/or individual with the best expertise in the matter, which results in a more efficient use of resources.

4. The History of FOSS

"The free/open source software movement began in the "hacker" culture of U.S. computer science laboratories (Stanford, Berkeley, Carnegie Mellon, and MIT) in the 1960's and 1970's.

The community of programmers was small, and close-knit. Code passed back and forth between the members of the community - if you made an improvement you were expected to submit your code to the community of developers. To withhold code was considered gauche--after all, you benefited from the work of your friends, you should return the favor."

4.1. A Brief History of Free/Open Source Software Movement^{vii}

The FOSS movement dates back to almost the very beginning of the computer industry, although it was not then formally defined or conceptualized. It was only in the late 1970s and early 1980s that the sharing of software began to really come in conflict with proprietary software. One of the earlier references to proprietary software was made by William H. Gates III in his now-famous "An Open Letter to Hobbyists."^{viii} In this letter, dated Tuesday, 3rd February 1976, he rails against the prevailing culture of software sharing:

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Proprietary software would gain momentum over the coming years. At the pioneering MIT Artificial Intelligence Lab in the early 1980s, a company called Symbolics was formed and took what was freely available code (the LISP programming language) and made it proprietary. In the process, it wiped out the software-sharing culture of the MIT lab at the time^{ix}. This destruction, however, would eventually result in the creation of the FSF and the FOSS culture today.

Richard Stallman, one of the MIT lab members at the time, was appalled at the turn of events. It would shape his view of proprietary software and instill in him the resolve to create a free operating system. The GNU (a recursive acronym for GNU is Not UNIX) project was born in January 1984. Over the next decade, it created a variety of critical tools that formed a portion of the operating system. The FSF was created a year later to promote free software and the GNU project. However, up until 1991, the GNU project had yet to produce a totally free software system due to a missing critical piece: the kernel.

The kernel is the heart of the operating system. In 1991, Linus Torvalds, who at the time was a second year graduate student at the University of Helsinki, wrote and distributed a UNIX-like kernel. In the manner of FOSS development, it was widely distributed, improved upon and soon adapted to become the core of the GNU/Linux operating system.

There were other FOSS projects in progress at the time, including BIND, Perl and the BSD operating systems. All of these projects eventually ended up merging or cross-pollinating.

The GNU/Linux operating system would continue to grow steadily in features and capabilities. In 1997, Linux exploded into the press limelight, with the International Data Corporation (IDC) noting that GNU/Linux already owned 25% of the server market^x and was growing at an annual compound growth rate of 25%.

In 1998, in response to Netscape's release of its Netscape Navigator code as FOSS, a group of FOSS developers came together, and the label "Open Source" was created. This led to the formation of the Open Source Initiative and the Open Source Definition. The primary purpose of this initiative was to get the corporate world to pay attention to the FOSS development process and steer a path away from the "confrontational" attitude of the free software movement^{xi}.

In 1999, the massively successful International Public Offering (IPO) of GNU/Linux distributor Red Hat gave it a market capitalization of US\$4.8 billion. Other successful IPOs that year were VA Linux (US\$ 7 billion), Cobalt Networks (\$3.1 billion) and Andover.net (\$712 million)^{xii}. As the poster child of FOSS, GNU/Linux's success meant that FOSS had truly arrived.

5. Why FOSS?

“Open-source software has been called many things: a movement, a fad, a virus, a Communist conspiracy, even the heart and soul of the Internet. But one point is often overlooked: Open-source software is also a highly effective vehicle for the transfer of wealth from the industrialized world to developing countries.”

Andrew Leonard, *“An Alternative Voice: How the Tech-Poor Can Still Be Software Rich”*^{xiii}

Is FOSS free?

The popular myth surrounding Free/Open Source Software is that it is always “free” — that is, “free of charge.” To a certain degree this is true. No true FOSS application charges a licensing fee for usage. Most FOSS distributions (Red Hat, SuSE, Debian, etc.) can be obtained at no charge off the Internet. On a licensing cost basis, FOSS applications are almost always cheaper than proprietary software.

Licensing costs, however, are not the only costs of a software package or infrastructure. It is also necessary to consider personnel costs, hardware requirements, opportunity costs and training costs. Often referred to as the Total Cost of Ownership (TCO), these costs give the clearest picture of the savings from using FOSS¹.

6. The Benefits of using FOSS

Besides the low cost of FOSS, there are many other reasons why public and private organizations are aggressively adopting FOSS. These include:

- Security
- Reliability/Stability
- Open standards and vendor independence
- Reduced reliance on imports
- Developing local software capacity
- Piracy, Intellectual Property Rights (IPR), and the World Trade Organization (WTO)
- Localization

Of particular importance to governments are the last four points as they are government-specific. Corporations and end-users usually do not deal with these issues.

6.1. Security

While there is no perfectly secure operating system or platform, factors such as development method, program architecture and target market can greatly affect the security of a system and consequently make it easier or more difficult to breach. There are some indications that FOSS systems are superior to proprietary systems in this respect:

- The Gartner Group recommends that businesses switch from Microsoft Internet Information Server (IIS) to Apache or another web server, due to IIS’s poor security track record. The Gartner Group noted that by July 2001 US enterprises had spent US\$1.2 billion simply fixing Code Red (IIS-related) vulnerabilities^{xiv}.
- “Hacker Insurance” issued by J.S. Wurzler Underwriting Managers costs five to 15 percent more if Windows is used instead of GNU/Linux or UNIX systems. Walter Kopf, senior vice president of underwriting at J.S. Wurzler Underwriting Managers, says, “We have found out that the possibility for loss is greater using the NT system.”^{xv}

The security aspect has already encouraged many public organizations to switch to, or to consider switching to, FOSS solutions. The French Customs and Indirect Taxation authority migrated to Red Hat Linux 6.2 largely because of security concerns^{xvi}.

Three reasons are often cited for FOSS’s better security record:

1 There is some argument that a better measure would be **Return on Investment (RoI)**. However, there are very few studies on the **RoI** of FOSS systems and it is just as difficult to measure as TCO, if not more. One article on **RoI** vs. TCO can be found at: http://www.infoworld.com/infoworld/article/03/08/29/34FElinux_1.html

- Availability of source code: The availability of the source code for FOSS systems has made it easier for developers and users to discover and fix vulnerabilities, often before a flaw can be exploited. Many of the vulnerabilities of FOSS listed in Bugtraq were errors discovered during periodic audits and fixed without any known exploits. FOSS systems normally employ proactive rather than reactive audits.
- Security focus, instead of user-friendliness: FOSS can be said to run a large part of the Internet^{xvii} and is therefore more focused on robustness and functionality, rather than on ease-of-use. Security considerations are taken into account before new features are added to any major FOSS application, and a feature is added only after it is determined that it will not compromise system security.
- Roots: FOSS systems are mostly based on the multi-user, network-ready UNIX model. Because of this, they come with a strong security and permission structure. Such models were critical when multiple users shared a single powerful server — that is, if security was weak, a single user could crash the server, steal private data from other users, or deprive other users of computing resources. Consequently, vulnerabilities in most applications result in only a limited security breach.

6.2. Reliability/Stability

FOSS systems are well-known for their stability and reliability. There are many anecdotal stories of FOSS servers functioning for years without requiring maintenance. Quantitative studies, however, are more difficult to come by. Here are two of the studies conducted to date:

- In 1999 Zdnet ran a 10-month reliability test between Red Hat Linux, Caldera Systems OpenLinux and Microsoft's Windows NT Server 4.0 with Service Pack 3. All three ran on identical hardware systems and performed printing, web serving and file serving functions. The result was that NT crashed once every six weeks, but none of the FOSS systems crashed at all during the entire 10 months^{xviii}.
- A stress test using random testing stressed seven commercial systems and the GNU/Linux system in 1995. Random characters were fed to these systems, to simulate garbage from bad data or users. The result was that the commercial systems had an average failure rate of 23%, while Linux as a whole failed 9% of the time. GNU utilities (software produced by the FSF under the GNU project) failed only 6% of the time. Years later, a follow-up study found that the flaws identified by the study were all fixed in the FOSS system, but were generally untouched in proprietary software^{xix}.

6.3. Open standards and vendor independence

Open standards give users, whether individuals or governments, flexibility and the freedom to change between different software packages, platforms and vendors. Proprietary, secret standards lock users into using software only from one vendor, and leave them at the mercy of the vendor at a later stage, when all their data is in the vendor's proprietary format, and the costs of converting them to an open standard is prohibitive.

The authors of the paper "Free/Libre and Open Source Software: Survey and Study" produced by the International Institute of Infonomics in the Netherlands also argue against the use of proprietary software in government. They say:

...Consequently one major argument against the implementation of proprietary software in the public sector is the subsequent dependency on proprietary software vendors. Whenever the proprietary standards are established the necessity to follow them is given. Even in an open tender acquisition system, this requirement for compatibility with proprietary standards makes the system biased towards specific software vendors, perpetuating a dependency.

Another advantage of FOSS is that it almost always uses open standards. This is due to two primary reasons:

- Availability of the source code: With the source code, it is always possible to reverse-

engineer and document the standard used by an application. All possible variations are plainly visible in the source code, making hiding a proprietary standard in FOSS systems impossible. Proprietary software, however, is much harder to reverse-engineer, and, in some cases, is deliberately obfuscated.

- **Active standards compliance:** When established standards exist, such as the HyperText Markup Language (HTML), which controls how web pages are displayed, FOSS projects work actively to follow the standards faithfully. The Mozilla web browser, a FOSS effort, is fully compliant with many standards from the World Wide Web Consortium (W3C). Webstandards.org notes that Mozilla is one of the most compliant browsers currently available today^{xx}. Compliance with standards is due to the FOSS development culture, where sharing and working together with other applications are the norm. It is also much easier to work with a globally dispersed group of developers when there is a published standard to adhere to.

Using FOSS systems as a means of gaining vendor independence has been raised in several areas. A report to the UK Government concludes that “the existence of an OSS reference implementation of a data standard has often accelerated the adoption of such standards, and recommends that the Government consider selective sponsorship of OSS reference implementations.”^{xxi}

6.4. Reduced reliance on imports

A major incentive for developing countries to adopt FOSS systems is the enormous cost of proprietary software licenses. Because virtually all proprietary software in developing countries is imported, their purchase consumes precious hard currency and foreign reserves. These reserves can be better spent on other development goals.

The European study, “Free/Libre and Open Source Software: Survey and Study”, also notes that, “The costs of this more service-oriented model of open source are then also normally spent within the economy of the governmental organization, and not necessary to large multinational companies. This has a positive feedback regarding employment, local investment base, tax revenue, etc.”^{xxii}

6.5. Developing local software capacity

It has been noted that there is a positive correlation between the growth of a FOSS developer base and the innovative capacities (software) of an economy. A report from the International Institute of Infonomics lists three reasons for this^{xxiii}:

- **Low barriers to entry:** FOSS, which encourages free modification and redistribution, is easy to obtain, use and learn from. Proprietary software tends to be much more restrictive, not just in the limited availability of source code, but also due to licensing, patent and copyright limitations. FOSS allows developers to build on existing knowledge and pre-built components, much like basic research.
- **FOSS as an excellent training system:** The open and collaborative nature of FOSS allows a student to examine and experiment with software concepts at virtually no direct cost to society. Likewise, a student can tap into the global collaborative FOSS development network that includes massive archives of technical information and interactive discussion tools.
- **FOSS as a source of standards:** FOSS often becomes a *de facto* standard by virtue of its dominance in a particular sector of an industry. By being involved in setting the standards in a particular FOSS application, a region can ensure that the standard produced takes into account regional needs and cultural considerations.

The FOSS developmental approach greatly facilitates not only innovation but also its dissemination. A Microsoft internal memo noted, “Research/teaching projects on top of Linux are easily ‘disseminated’ due to the wide availability of Linux source. In particular, this often means that new research ideas are first implemented and available on Linux before they are available/incorporated into other platforms.”^{xxiv}

6.6. Piracy, Intellectual Property Rights (IPR), and the World Trade Organization (WTO)

Software piracy is a problem in almost every country around the world. According to the Business Software Alliance-IDC Global Software Piracy Study, software piracy in 2005 cost US\$34 billion in global losses. Even in developed nations where software is affordable in theory, piracy rates were as high as 21% in the United States and 36% in the European Union. Among developing countries, where lower incomes make software far more expensive, Pakistan's piracy rate is 86%.^{xxv}

Software piracy and lax laws against it can and does hurt a country in many ways. A country with poor protection for Intellectual Property Rights (IPR) is unattractive to foreign investors. Membership in the World Trade Organization (WTO) and access to its benefits are strongly affected by the level of protection given to IPR in a country. Finally, a culture of software piracy hurts local software development, as there is less incentive for local software developers to create a local product.

6.7. Localization

“Localization involves taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold.”

Localisation Industry Standards Association^{xxvi}

Localization is one of the areas where FOSS shines because of its open nature. Users are able to modify FOSS to suit the unique requirements of a particular cultural region, regardless of its economic size. All that is necessary is the technical capability within a small number of individuals to create a minimally localized version of any FOSS. While the construction of a completely localized software platform is no small feat, it is at least possible. Most initial FOSS initiatives in the Asia-Pacific region have dealt with localizing FOSS. Microsoft's decision in 1998 against producing an Icelandic version of Windows 98^{xxvii} would have had serious implications if it were not for the emergence of FOSS alternatives.

7. The Shortcomings of FOSS

For all its benefits, FOSS is not suitable for every situation. There are areas where FOSS needs improvement.

7.1. Lack of business applications

While there are many FOSS projects currently in various stages of development, there are still many areas that lack a full-featured product, especially in the business world. The recent porting of Enterprise Resource Planning platforms such as SAP and Peoplesoft^{xxviii} have helped cover the high-end application market, but the Small and Medium Enterprise (SME) market is still poorly served. Basic, polished accounting applications such as Quickbooks, Peachtree or Great Plains do not have FOSS equivalents at this time.

This problem has come about in part due to the scarcity of people who are competent in both technical and business subjects. Technical developers who encountered problems and wrote software to “scratch an itch” started most of the existing FOSS projects today. These projects are usually fairly technical in nature, such as the creation of web servers, programming languages and environments, and networking tools. It is rare for a software developer to encounter accounting problems, for example, and, at the same time, have the business knowledge required to create a technical solution.

7.2. Interoperability with proprietary systems

FOSS systems, especially on the desktop, are not completely compatible with proprietary systems. For organizations that have already invested massive amounts of capital into proprietary applications and data storage formats, attempting to integrate FOSS solutions can prove to be prohibitively expensive. Changing proprietary standards, which is often aimed at preventing the integration of alternate solutions, exacerbates this problem.

In time, as organizations shift from proprietary to open standards, this problem should be reduced.

7.3. Documentation and “polish”

Established FOSS lacks the extensive documentation and user-friendliness found in commercial software^{xxix}. The primary focus of early FOSS developers was functionality. Creating a program that worked well was far more important than ease-of-use.

Besides the dearth of high-quality documentation, there are also user interface issues with FOSS Graphical User Interfaces (GUI). Because the GUI element in most FOSS systems is not a single element but a collection of different projects glued together, the behavior of the GUI elements differ greatly. Command-to-save data differ from one program to another, quite unlike proprietary desktop operating systems, such as the Mac OS X or Microsoft Windows. Cutting and pasting between different programs can be wildly inconsistent or even impossible. While there is significant ongoing work to unify the desktop, the desktop is likely to remain inconsistent for some time to come.

8. The Licensing Arrangements for FOSS - An Overview of Intellectual Property Rights

Nowadays, the intangible products of human beings' creative activities are considered to be a kind of property and protected as their tangible counterparts. The idea of intellectual property rights has been generally accepted, and legal institutes are built to offer protection to their possessors. Although copyrights, patents, trademarks and trade secrets all fall into the greater category of intellectual property, the essence of each differs from the other.

- Trade Secrets: A trade secret is protected to avoid being accessed by its owner's competing business entities. This can be done through a variety of civil and commercial means, such as confidentiality agreements or non-disclosure agreements signed by those who are given accesses to such special knowledge and information.²
- Trademarks: Trademarks are the distinctive names, phrases, symbols, designs, pictures or styles used by a business to identify itself, and its products or services, to its consumers. In many countries, colors, three-dimensional marks, sounds and even smells are also eligible for trademark protection.³
- Patents: While trade secrets promote the competency of a business by withholding certain information from the public, patents are designed to provide the inventor with the monopoly over newly-developed knowledge for a certain period of time (usually 20 years) in exchange for its disclosure. Typically, in order to gain such exclusive rights, the inventor is required to file a patent application, which is then reviewed by a designated patent examiner. The novelty of the invention is an essential element in granting a patent.⁴
- Copyright: Copyright is applied to various kinds of human creative works, such as literary works, music compositions, paintings and software. The copyright holder of a work is entitled to exclusive rights of the reproduction, modification, distribution and public display of the work. Unlike patent protection, copyright law is applied to a work upon its creation, regardless of its novelty. The ideas embraced in the work are not protected; copyright only prevents others from copying the copyright-holder's particular way of expressing the ideas.⁵

8.1. How is software protected?

Software, like other literary works, is now protected under copyright law. Although in recently years it has been argued that the source code and algorithms should be patentable, and have already been granted patents in some case, software patents are still questioned and contested by many, especially from the FOSS community. This tutorial is focusing only on FOSS licenses; the software patent is itself too complicated an issue and will not be addressed much here.

2 Available from http://en.wikipedia.org/wiki/Trade_secret, Internet; accessed 27 July 2004

3 Available from <http://en.wikipedia.org/wiki/Trademark>, Internet; accessed 27 July 2004

4 Available from <http://en.wikipedia.org/wiki/Patents>, Internet; Accessed 27 July 2004

5 Available from <http://en.wikipedia.org/wiki/Copyright>, Internet, Accessed 27 July 2004

8.2. Copyright Basics

8.2.1. What can be copyrighted?

Copyright protects the expressions of ideas in different forms of works, including artistic, dramatic, literary, musical, and other intellectual works.⁶ Since the 1980s, a software program, like other literary works, has been protected under copyright law.⁷ The ideas expressed in such works themselves are not protected.⁸

8.2.2. Is anything required to be done in order to get a work protected?

Nowadays, copyright law does not require formalities to gain copyright protection. You, the author, do not need to publish, register, or pay registration fee of any kind, and do not need to have a copyright notice attached to it. The copyright of the work immediately becomes your property, the person who created the work. Copyright protection is automatically applied to a work once it is created.⁹

8.2.3. What are the rights granted to the copyright-holder?

“Copyright” is a set of exclusive legal rights granted by governments to the author who created a work. This includes the right to reproduce the work, the right to prepare derivative works based upon the work, the right to distribute copies of a work, the right to perform and display a copyrighted work publicly, and some other rights defined in copyright law.¹⁰ Without an explicit expressed consent from you, the copyright-holder, it is illegal for anyone to violate any of your rights.

Copyright law protection has been expanded largely with time.

8.3. The Expansion of Copyright Protection

8.3.1. The first copyright legislation (Statute of Anne, 1710)

Compared to other legal institutions, copyright law came relatively late in human civilization. The first known copyright legislation was the Statute of Anne, enacted in 1710, in Great Britain.¹¹ For a newly created work, the Statute of Anne protects the copyright-holder’s right to print and reprint the books and other writings for 14 years.

8.3.2. All-dimensional expansion of copyright protection

In its initial stages, the Statute of Anne’s scope of copyright protection is quite limited. The works protected are limited to “books and other writings”, the rights granted to the copyright holder are limited to the “print” and the “reprint” of the work, and the length of the protection is limited to “14 years”.

Nowadays, copyright law protects much more than that. The works protected now include architecture, music compositions, music recordings, paintings, sculpture, and software. The rights granted to the copyright-holder now expand to the printing, reprinting, modification, public

6 Available from <http://www.wipo.int/copyright/en/faq/faqs.htm#rights>, Internet, Accessed 28 June 2004

7 Available from http://www.wipo.int/copyright/en/faq/faqs.htm#P39_5114, Internet, Accessed 28 June 2004

8 Available from <http://www.wipo.int/copyright/en/faq/faqs.htm#ideas>, Internet, Accessed 28 June 2004

9 Available from <http://www.wipo.int/copyright/en/faq/faqs.htm#rights>, Internet, Accessed 28 June 2004

10 Richard, Jones, “Copyright Protection for Computer Software in the United States”, 2002, Available from <http://www.ladas.com/Patents/Computer/SoftwareAndCopyright/Softwa02.html>, Internet Accessed on 28 June 2004

11 Available from <http://www.copyrighthistory.com/anne.html>, Internet, Accessed on 28 June 2004

display, public performance, and distribution of the work. And the life of the copyright has been increased to 50 years after the author has died. (In Europe and the in the US, it has been expanded to 70 years.)¹²

8.4. From national to international protection

The expansion of copyright protection does not affect copyright law in one jurisdiction alone; it has now become an international standard worldwide.

8.4.1. Berne Convention

In the late 19th century, while copyrighted works gradually became an important item in international trade, trans-national copyright protection gradually became a serious issue. Beginning with its European signatories, the Berne Convention 1886 first introduced the national treatment principle. While the Berne Convention requires its signatories to uphold basic copyright protection, the national treatment principle protects the work of a foreign copyright-holder in the same way as it protects the work of its own nationals. The Berne Convention has, therefore, created an international standard for copyright protection.

Without a dispute resolution mechanism, however, the protection that the Berne Convention offers is relatively weak. It is too costly for you, if you are a copyright-holder, to claim your rights in a foreign country, in which you believe that your rights have been infringed upon.

8.4.2. A more enforceable international standard - WTO and TRIPs

In the 1990s, the World Trade Organization (WTO) and its Trade Related Intellectual Property Agreements (TRIPs) have become a more powerful structure for international copyright protection. Every economy that intends to become a member of the WTO is required to sign the TRIPs, and every TRIPs signatory must agree to comply with all of the key sections of the Berne Convention. The WTO also provides a dispute-settlement and enforcement mechanism for copyright infringement among member countries. International copyright protection has now become a more enforceable standard.¹³

8.4.3. Protection upon creation, the abolishment of formality requirements

As a principle set forth by the 1908 Berne Convention, copyright protection is applied to a work once it is created without requiring any formality.¹⁴ You, the author, do not need to register, nor even publish a work, in order to enjoy full-fledged copyright protection. Even though the scope and terms of copyright law in different countries might have changed with time, but for Berne Convention signatories, unless otherwise expressed explicitly by you, the law assumes that you claim all kinds of rights granted to you.

With the standardization of international copyright protection, copyright laws in different countries have been revised to comply with such standards. In anticipating joining the Berne Convention Union, for example, the US revised its Copyright Act and abolished formality requirements in 1976.¹⁵

8.4.4. Copyright law, a balance between public and private interests

When compared to other legal institutions, copyright protection is a relatively new invention in human history. With time, the development of copyright regulation also reflects the social and technological transformation regarding human creative activity and its distribution. While granting exclusive private rights to authors or copyright-holders has been considered a means

12 Little, Jonathan, "History of Copyright- A Chronology", 2002, Available from <http://www.musicjournal.org/01copyright.html>, Internet, Accessed on 28 June 2004

13 Story, Alan, "Don't Ignore Copyright, the 'Sleeping Giant' on the TRIPs and International Educational Agenda", pp.132-33, in Global Intellectual Property Rights, Knowledge, Access and Development, Drahos, Peter and Ruth Mayne Eds., NY: MacMillan, 2002

14 Lessig, Lawrence, "Free Culture", Footnote 194, Available from <http://www.jus.uio.no/sisu/freeculture.lawrence.lessig/14>, Internet, Accessed on 29 June 2004

15 Little, Jonathan, "History of **Copyright - A** Chronology", 2002, Available from <http://www.musicjournal.org/01copyright.html>, Internet, Accessed on 28 June 2004

to promote human creative activity; copyright law also bears the recognition of the larger public interest, particularly with regard to education, research and access to information.¹⁶

Various measures have been adopted in order to achieve a balance between public and private interests. In the Statute of Anne, the law states that the authorities can limit and settle the price of printed books according to their best judgment. In the US Constitution, under Congressional decision, exclusive rights to their writings and discoveries can be granted to authors and inventors within a limited timeframe. While recognizing that the exclusive rights granted by the law might hinder the public accessibility of information and knowledge, certain limits, such as fair use and first sale doctrine, are imposed upon such exclusive private rights during its limited lifetime.

8.5. Software and Copyright Protection

8.5.1. The extension of copyright law protection to software in the 1980s

The US Computer Software Copyright Act 1980 regards computer programs as works that can be protected by copyright law. Since then, it has become an international trend that copyright protection can be applied to computer software as well. The WIPO Copyright Treaty (1996) also states that computer software is eligible for protection by copyright law.

8.5.2. Copyright protects both the source and the object code under TRIPs

Software can be expressed in both the source code and the object code. While the ideas expressed in the form of source code can be perceived by trained programmers, they cannot be understood by human beings when they are expressed in the form of object code. Since it is stated in the TRIPs that copyright protection of computer software applies to both source code and object code forms,¹⁷ so, in practice, proprietary software companies tend to release their product only in object forms, and keep the source code of the product as their trade secret.¹⁸

As stated earlier, copyright law protects only the expressions of ideas, but not the ideas themselves. Upon receiving a copy of a literary work, or a musical composition, the ideas expressed in the work can be perceived, and might become an inspiration for other new works, thereby contributing to human intellectual development. But when software, distributed only in the form of object code can also receive copyright protection, this means that the proprietary company can enjoy full-fledged copyright protection without sharing its ideas. The knowledge gained in the making of the software is inaccessible even to the trained developer, let alone the general public. Copyright protection regarding software in such cases is, therefore, not in line with the essence of copyright law, i.e. achieving a balance between private and public interests.

8.5.3. Users' rights disguised in proprietary licensing models

Under traditional proprietary licenses, the source code is inaccessible. Proprietary licenses might even forbid developers from studying the program. In the licenses for developers, for example, such as the Microsoft End User Agreement, and the Microsoft Developer Network Subscription, reverse-engineering, recompilation and disassembly are not allowed, except and only to the extent that it is expressly permitted by the applicable law.¹⁹

For end-users, proprietary licenses usually allow only one copy for each computer or each processor. That means that if you have one desktop and one laptop, or two desktops, you will have to purchase two copies if you want to run the program legally on both your machines. When there are bugs in the program that you have legally purchased, since you do not have access to the source code, and are not allowed to study the program, you will remain vulnerable and passive to notifications by the proprietary companies or hackers regarding the

¹⁶ As stated in the Preamble of WIPO Copyright Treaty, Available from <http://www.wipo.int/clea/docs/en/wo/wo033en.htm>, Internet, Accessed on 29 June 2004

¹⁷ Available from http://www.wto.org/english/docs_e/legal_e/legal_e.htm#TRIPs, Internet, Accessed on 28 June 2004

¹⁸ Halligan, R. Mark, "How to Protect Intellectual Property Right in Computer Software", Available from <http://my.execpc.com/~mhalign/computer.html>, Internet, Accessed on 1 July 2004

¹⁹ Available from <http://www.msdnaa.net/EULA/NA/English.aspx>, Internet, Accessed 04 Aug 2004

existence of such bugs. Even if you are notified, you will be unable to legally debug anything on your own, or use unofficial patches, since modifying the program is not granted either.

On the other hand, the FOSS movement has contributed to a positive transformation of such a situation. As stated by the Free Software Foundation (FSF) when it was founded in 1985, it is dedicated to promote the users' rights to use, study, copy, modify, redistribute computer programs,²⁰ the rights to which are largely disguised, if not ignored, in traditional proprietary licensing models.

9. How is FOSS different from proprietary software?

The development of FOSS might be taken as a reaction by the software developers' community to the existing legal instruments regarding software copyright. Both consider the access to the source code as a pre-requisite, and go further to other rights bundled in copyright, such as the right to make copies of a work, the right to distribute the copies of a work, and the right to prepare derivative works from a work.

9.1. Free Software

Background: Transition in the Information Technology Industry and Legal Institutions

In the 1970s, the transformation in legal institutions and in the Information Technology (IT) industry contributed to the formation of free software. The US revised its copyright law, and IT companies also continued to claim software as copyrightable work.²¹ On the other hand, while earlier software was bundled with hardware in the hardware market, the IT industry started to consider software as a separate market.²² It started to recruit more developers from laboratories in research institutes, and asked them to sign confidential agreements.

9.2. Richard Stallman on a stark moral decision

Prior to that, the common practice in laboratories was largely about sharing sources and copies. For Richard Stallman, who worked in the Massachusetts Information Technology (MIT) laboratory at that time, such a transformation was accompanied by the diminishing of the community which honored sharing and the ethic of "helping your neighbors", a community he was proud to belong to. As a talented programmer who could easily sign a contract and a confidential agreement with a proprietary company in exchange for a well-paid salary, Stallman said that he took a "stark moral decision". He had to decide whether to join the software industry, or to struggle for the survival and sustainability of the community to which he belonged. He chose the latter, and developed the necessary infrastructure to maintain the free software community.²³

9.3. Free Software Definition

Free software is about granting the users' freedom to run, copy, distribute, study, change and improve the software. Stallman developed the definition of free software given below:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.²⁴

20 Available from <http://www.fsf.org/fsf/fsf.htm>, Internet, Accessed 04 Aug 2004

21 Richard, Jones, "Copyright Protection for Computer Software in the United States", 2002, Available from <http://www.ladas.com/Patents/Computer/SoftwareAndCopyright/Softwa04.html>, Internet, Accessed on 28 June 2004

22 Campbell-Kelly, Martin, "Development and Structure of the International Software Industry, 1950-1990", Available from <http://www.dcs.warwick.ac.uk/~mck/Personal/SoftIndy.pdf>, Internet, Accessed on 1 July, 2004

23 Stallman, Richard, "The GNU Operating System and the Free Software Movement", pp.53-56, CA: O'Reilly & Associates, Inc., 1999.

24 Available from <http://www.fsf.org/philosophy/free-sw.html>, Accessed on 31 May 2003

Other than emphasizing the access to the source code, the Free Software Definition also pointed out the right of the user to copy, to distribute the copy, to modify the software, and to distribute the derivative work of a copyrighted work. (The previous three kinds of rights are granted exclusively to its copyright-holder by the existing copyright law).

10. Creating a Free Software Environment

10.1. The GNU Project and the Free Software Foundation

The Free Software Definition strongly claims the rights of non-copyright holders. But without the environment that allows such an ideal, the above freedoms would merely be unrealistic slogans. The GNU project was launched in 1984 to develop and complete a UNIX-style operating system which is free software, i.e., the GNU system, and the project went further to include other application software.

In 1985, the Free Software Foundation (FSF) was established to promote the idea of free software. The FSF promotes the development and use of free software by distributing the software which happens to be available, and also by concentrating on the development of free software, aiming at creating a coherent system, the GNU operating system, and providing alternative solutions to proprietary software. For further details, please visit <http://www.gnu.org/> and <http://www.fsf.org/>.

10.2. GNU General Public License (GNU GPL or GPL)

Under the existing legal structure, once the work is created, copyright protection is granted to the exclusive disposal of the copyright-holder. Without an explicit expression, it is assumed that you, the copyright-holder, claim all the rights granted to you, and that any dissimilar subject opinion must be ignored. That is to say, the law burdens you with an explicit expression not to claim some or all rights granted to you.

While people might sometimes not know how to make such an explicit expression, the creation of the GNU General Public License (GNU GPL or GPL) serves as a legal tool to bring out the freedoms stated in the Free Software Definition and to maintain the environment that supports free software.

The GPL is a license, but it is different from proprietary licenses. It grants the users the rights that were considered the exclusive right of the copyright-holder by law and by business practices, including the right to access to the source code, the right to run the program, the right to make copies and redistribute the copies, and the right to modify the program and distribute the modified program. On the other hand, although the GPL grants the users many rights and freedoms to use the software, it also sets certain limitations in order to ensure that free software, and its derivations, remains as free as it is.²⁵

When a work is licensed under GPL, it means that you, the author of the software, still claim copyright of the work, but can adopt a different license as a way of explicit expression to allow the public to have greater freedom to use your work than copyright law assumes.

The characteristics of the GPL will be further explained in a separate section below.

11. Open Source Software

While advocates of free software consider the freedom of software a moral issue, some promote the idea of “Open Source Software”, which focuses more on the technical values of FOSS and is more business-friendly.²⁶ The Open Source Initiative (OSI) operates as an organization to promote the open source campaign by managing and promoting the Open Source Definition and its certification mark for open source licenses and products.

²⁵ See GPL Preamble, Available from <http://www.fsf.org/licenses/gpl.txt>, Internet, Accessed on 31 May 2004

²⁶ Wong, Kenneth and Phet Sayo, “Free/Open Source Software, A General Introduction”, pp. 6-7, 2004, Available from <http://www.iosn.net/index.php?module=ContentExpress&func=display&ceid=95>, Internet Accessed on 31 May 2004

11.1. Open Source Definition

The Open Source Definition is a revision from a policy document of the Debian GNU/Linux Distribution, which served to clarify which licenses are free licenses.²⁷ The OSI explains the basic idea behind open source:

“The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs.”

Open source also emphasizes the same rights stated in the Free Software Definition, including the users’ access to source code, users’ right to copy the work, distribute the copies, modify the work and distribute the derivative works.

Compared to the Free Software Definition, the Open Source Definition, which consists of 10 clauses, is relatively lengthy. Besides the clauses regarding the accessibility of the source code (section 2), users’ right to copy and distribute the original work (section 1), users’ right to modify the work and distribute the derivative work (section 3), the Open Source Definition also has several non-discrimination clauses (sections 5, 6, 8, 9, 10). Although not stated in the same way, such non-discriminatory ideas can also be found in the Free Software Definition. Section 7 of the Open Source Definition aims to prevent the source from being closed again, which is a central concept in free software. The emphasis on the integrity of the author’s source code and the requirements regarding the distribution of modified works (section 4) is not explicitly stated in the Free Software Definition.

<http://opensource.org/docs/definition.php>

12. OSI-approved Licenses

With the guidelines stated in the Open Source Definition, the OSI certifies and recognizes licenses as Open Source Licenses after following certain procedures. The certification procedure is upon request, and if it is a newly-approved open source license, it will be added to a list of open source licenses maintained by the OSI.

http://opensource.org/docs/certification_mark.php

The number of OSI-approved licenses also grows with FOSS development in different areas. Licenses derived from the FOSS community, e.g. GPL, Lesser General Public License (LGPL), PHP License, Nethack General Public License; those from academic and research institutes, e.g. NASA Open Source Agreement, MIT License, University of Illinois/NCSA Open Source License; and those from proprietary companies, which have adopted FOSS as part of their strategies, e.g. Apple Public License, Eclipse Public License, Qt Public License, Mozilla Public License. A large proportion of the OSI-approved licenses are from proprietary companies.

12.1. Free or Restrictive?

Although at first sight the Free Software Definition and the Open Source Definition don’t seem to be all that different from one another, they do differ in some rhetoric, which also reflects their differences in philosophy.

<http://www.fsf.org/philosophy/categories.html>

Some people, for example, might refer to the classic free software licenses such as the GPL and the LGPL as “highly restrictive” and “restrictive”; for they have set up many restrictions to make sure that the free software and their derivative works stay free. But for the FSF, these restrictions are pre-requisites for a healthy environment for free software, therefore the free software licenses are not supposed to be referred to as “highly restrictive” or “restrictive licenses”.

The FSF also maintains a list of free software licenses and non-free software licenses. Although the FSF might sometimes refer to those relatively simple licenses as “permissive”, it never refers to the more complicated free licenses as “restrictive”.

Despite such philosophical differences, however, both the FSF and the OSI agree with each other in most cases regarding which license may be classified as a FOSS license.

²⁷ Perens, Bruce, “The Open Source Definition”, in Open Sources, Voice From the Open Source Revolution, p173, CA: O’Reilly & Associates, Inc., 1999.

Among the 26 OSI-approved licenses analyzed by the FSF, only two, the Original Artistic License and the Reciprocal Public License, are regarded as non-free licenses (see table given below).

12.2. How to make the source free/open?

As explained above, under current legal institutions, like other literary works, software is also protected by copyright law. The FOSS movement makes use of legal institutions. It makes you - the software's author - grant the users some of the rights which are granted exclusively to you by copyright law. The creation of FOSS licenses help you make such an announcement easily, and also form as agreements of the FOSS developers' communities.

There are many FOSS licenses, and each differs in characteristics. We will also discuss the three major kinds of licenses: the GNU GPL, the LGPL and the BSD-style License. They represent the three very different styles of FOSS licensing, and are also the most pervasively adopted licenses.²⁸ But first, for a quick and general overview, observe the following table developed by the Open Source Software Foundry (OSSF) Supports Software Freedom http://www.openfoundry.org/index.pl?section=en_index.²⁹

28 If we look at the SourceForge.net, the largest FOSS development website, we can see GPL, LGPL and BSD are the 3 most adopted licenses. Among all 53026 projects that is licensed under OSI-approved licenses, 36962 projects are licensed under GPL, 5817 projects are under LGPL and 3813 projects are licensed under BSD. Available from http://sourceforge.net/softwaremap/trove_list.php?form_cat=14, Internet, Accessed 1 Aug 2004

29 Open Source Software Foundry (OSSF) Supports Software Freedom, A Comparison of FOSS Licenses v2.1, Available from http://www.openfoundry.org/index.pl?section=en_index, Internet, Accessed 26 May 2006

12.3. A Comparison of FOSS Licenses v2.1

Original Program				
	Do you require that source code be given when the program is redistributed?	Do you agree to let your software be sublicensed by your licensees?	Do you allow other people to use the patent involved in your program?	If the distributed program does not include source code and the source code is provided through other means, do you allow that the fee charged for the distributed source code can exceed the cost of distributing the source code?
MIT	N	Y	N	Y
BSD	N	N	N	Y
zlib/libpng	N	N	N	Y
Apache 1.1	N	N	N	Y
Apache 2.0	N	Y	Y	Y
Artistic	Y	N	N	Y
CPL 1.0	Y	Y	Y	Y
QPL 1.0	Y	N	N	N
MPL 1.1	Y	Y	Y	Y
LGPL 2.1	Y	N	N	N
GPL 2.0	Y	N	N	N

Modified Program

	Do you allow the modified work of your program to be licensed under different terms?	Continued from the last question. Under which conditions do you allow the licensee who modifies the program to adopt a license of his or her own?	Do you require that the licensee who modifies your code supply complementary documentation?	Do you require the licensee who modifies your code supply complementary documentation?
		0. The licensee can only adopt the same license. 1. The licensee can choose any license he or she likes, even if the license is not a free software one. 2. The licensee can only adopt a license that I designate. 3. The licensee can decide which terms he or she likes to use so long as they do not contradict the license that I have adopted.		
MIT	Y	1	N	N
BSD	Y	1	N	N
zlib/libpng	Y	1	N	Y
Apache 1.1	Y	1	N	Y
Apache 2.0	Y	3	N	Y
Artistic	Y	1	Y	Y
CPL 1.0	Y	3	Y	N
QPL 1.0	Y	3	Y	N
MPL 1.1	Y	2	Y	Y
LGPL 2.1	N	0	Y	Y
GPL 2.0	N	0	Y	Y

All the FOSS licenses discussed in the above-mentioned table, by definition, contain the following common features:

- Source code of the original work should be open
- It is allowed to make copies of the original work
- Distribution of the original work is allowed, and copyright notice should be attached in all distributions
- The license grant is non-exclusive, global, royalty-free, and for all purposes
- Warranty is disclaimed

Although certain rights have to be granted to the users, these FOSS licenses differ from each other regarding how these rights may be exercised. For example, although the offering of access to the source code by you, the author, is essential for FOSS licenses, whether redistributors are also required to do so varies between different licenses. When redistributing a BSD-ed program, for example, you are not required to provide the source code. For those who received the program directly from you, the source code of the original will still be accessible, since it is offered by you.

And even when redistributors are also obliged to provide the source code, there are also different regulations regarding the distribution fee the redistributors can collect. The GPL and the LGPL are particularly detailed about when a fee higher than the physical transmission fee may be collected. This is because the GPL and the LGPL provide redistributors various ways to distribute the program, whether with, or without, the source code. But despite such freedom, the GPL and the LGPL also wish to keep the acts of redistribution in line with the spirit of free software. You can sell free software for any sum of money you wish, since the market will help to keep it within a reasonable range; but if the package is sold without the source code, then the fee collected for the distribution of source code itself cannot exceed the physical transferring cost.

Clauses regarding derivative works vary to a greater extent. Although the source code of original works is required by definition, the source code of derivative works is not. Even if a FOSS license requires the source code of derivative works to be open, it might not have to be licensed under exactly the same license as the original work is. For example, although a derivative work of a GPL-ed program also has to be licensed under the GPL, a derivative work of a BSD-ed program does not have to be licensed under the BSD. In fact, a derivative work of a BSD-ed program doesn't even have to open its source.

It is also in this respect that FOSS licenses differ with each other regarding the possibility of allowing a FOSS program to be combined with proprietary programs. When combining different programs into a larger project, it is quite inevitable that the larger project, while embracing all, or part of, the source code of the programs it combined, becomes the derivative work of all the combined programs. For example, project ABC is combined with a GPL-ed program A, a BSD program B and a proprietary program C, and has source code from all three programs. Although, when being a derivative work of program B, project ABC is not required to be licensed either under the BSD License or even open its source, it is, however, as a derivative work of program A, required to be licensed under the GPL. You, the developer, will have no choice other than to license the project ABC under the GPL, or find an alternative solution to program A, especially if you wish to make it a proprietary software project.

It is in this sense that the GPL is considered to have a so-called "viral" effect, and is unfriendly to proprietary software development. It is also in this sense that the GNU Lesser General Public License (LGPL) is designed to encourage more use of free libraries.

The three typical FOSS licenses, the GPL, the LGPL and the BSD are explained in greater detail below:

12.4. GNU General Public License (GNU GPL or GPL)

The GNU General Public License (GNU GPL or GPL) is the classic free software license. It is also the most well-known and most widely adopted among all FOSS licenses. The GPL is an invention brought out to fulfill the freedoms defined by free software. It is both a license and a document to manifest the basic idea behind free software.

12.4.1 Copyleft

The way in which GPL guarantees this freedom is also called “copyleft”. While traditional proprietors say “Copyright, All Rights Reserved”, it has been imaginatively rephrased as “Copyleft, All Rights Reversed”.

Copyleft prevents free software from being turned into proprietary software again. It uses copyright law, but serves the opposite of its usual purpose. Instead of becoming a means of privatizing software, copyleft uses rights granted to authors to keep the software free.³⁰

Instead of being a work in the public domain that everyone is free to make use of, a GPL-ed work or a copyleft-ed work is still a copyrighted work. You, the author of the GPL-ed work, do not give up your rights as a copyright-holder; instead, you exercise your rights in a way which is different from traditional proprietors.

You, as an author who wants to make your software free, cannot just disclaim your rights as the copyright-holder, and release the work into the public domain, for that would mean exposing your work to the danger of being privatized again. Instead, you have to claim your rights, and with the help of those exclusive rights, be able to regulate the ways in which people make use of your work. By licensing your work under the GPL, you will be allowing the users to have the rights allowed by the free software movement, and also ask the users to take on some responsibilities to keep the software and its derivative works as free as the way you intended your work to be.

12.4.2. Major terms and conditions of the GPL

Users' freedoms

When a program is licensed under the GPL, besides the access to the source code, users are free to:

- Run the program (section 0)
- Make copies of the program (section 1)
- Redistribute the program, even for commercial purposes, provided an appropriate copyright notice and disclaimer of warranty are retained (section 1). Redistribution in the object code or executable form is also possible, so long as source code is available for all recipients (section 3)
- Prepare and distribute derivative works of the program, provided the derivative works are also licensed to all third parties under GPL (section 2)

No warranty

Although the distribution of the work may be commercial, the work itself is licensed free-of-charge. There is, therefore, no warranty for the GPL-ed software (sections 11, 12). The distributor can choose to offer warranty protection in exchange for a fee (section 1).

License issued directly from the author

The work is not sub-licensable. When a program is redistributed, the recipients still receive the license from the original licensor, and the redistributors may not impose any further restrictions on the recipients' exercise of the rights granted in the GPL (section 6).

Acceptance and termination

By modifying or distributing the GPL-ed program, a person indicates his or her acceptance of the license (section 5). The license grant is irrevocable, but when the licensee violates the license, the rights granted will be terminated automatically. But those parties who received the program from him or her, since they all received the license from the original licensor, their rights will not be affected so long as they remain in full compliance with the license (section 4).

Co-existence with other legal obligations?

The GPL does not concede to any contradictory conditions that are imposed on the recipients. Compliance with the license, if it is not possible as a consequence of a court judgment, an

30 Stallman, Richard, “The GNU Operating System and the Free Software Movement”, p.59, CA: O’Reilly & Associates, Inc., 1999.

allegation of patent infringement, or for any other reason, the recipient may not redistribute the program at all (section 7). A GPL-ed program cannot be incorporated into a proprietary program, nor can it be linked with a proprietary library.

The full text of the GPL is available at <http://www.gnu.org/licenses/gpl.html>

The FSF also maintains a FAQs section about the GPL, which can be accessed at <http://www.fsf.org/licensing/licenses/gpl-faq.html>

12.5. GNU Lesser General Public License (GNU LGPL or LGPL)

Aside from the GPL, the GNU project offers a special kind of copyleft license for libraries. The GNU Lesser General Public License (LGPL) permits LGPL-ed libraries to be linked with proprietary software.

Such an exception may be found in different situations. It might be a strategic decision in order to encourage proprietary applications on the GNU system.³¹ And for a free library, whose features can largely be replaced by other proprietary features, release under the LGPL rather than the GPL encourages its wider use,³² which helps to make more improvements in it, and, with a larger body of free software users, garners wider support for free software.³³

Free software advocates, however, still encourage people to use the GPL for their libraries rather than the LGPL, especially for those libraries which possess significantly unique capabilities. People who are interested in utilizing such GPL-ed libraries will have to release their modules as free software as well, resulting in more useful modules and program being developed within the free software environment.³⁴

12.5.1. Major terms and conditions of the LGPL

The LGPL is identical to the GPL in many ways, including the clause that disclaims warranty, which states that the license is issued directly from the author, when the license is to be applied and terminated, and its relationship to other legal obligations applied upon the users.

But when it comes to users' rights, the LGPL distinguishes between two different kinds of situations. When one uses a library, a "work based on the Library" means either the Library itself or any derivative work under copyright law (section 0), while a "work that uses the Library" means a program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it (section 5).

Works based on the Library

In the case of a "work that uses the Library", i.e. the Library itself and its derivative works, the terms are very similar to those outlined in the ordinary GPL.

User's Freedoms

- Run the program (section 0)
- Make copies of the program (section 1)
- Redistribute the program, even for commercial purposes, provided an appropriate copyright notice and disclaimer of warranty are retained (section 1). Redistribution in the object code or executable form is also possible, so long as source code is available for all recipients (section 4)
- Prepare and distribute derivative works of the program, provided the derivative works are also licensed to all third parties under LGPL (section 2c)

31 Stallman, Richard, "The GNU Operating System and the Free Software Movement", p.63, CA: O'Reilly & Associates, Inc., 1999.

32 Stallman, Richard, "Why you shouldn't use the Library GPL for Your Next Library". Feb 1999. Available from <http://www.fsf.org/licenses/why-not-lgpl.html>, Internet, Accessed 29 May 2004

33 Preamble, the "GNU Lesser General Public License" Available from <http://www.fsf.org/licenses/lgpl.txt>, Internet, Accessed 29 May 2004

34 Stallman, Richard, "Why you shouldn't use the Library GPL for Your Next Library". Feb 1999. Available from <http://www.fsf.org/licenses/why-not-lgpl.html>, Internet, Accessed 29 May 2004

In addition, you may also opt to apply the terms of the GPL instead of the LGPL to a given copy of the LGPL-ed library, especially when you are incorporating part of the code into a program that is not a library (section 3).

Works that use the Library

In the case of a “work that uses the Library”, the work itself is not subject to the LGPL. But when linking the “work that uses the Library” with the Library, an executable which is a derivative work of the Library will be created, and such an executable is covered by the LGPL (section 5).

Such executables, although the LGPL allows you, the author, to distribute the object code (section 5) and license it under the terms of your choice, it also requires that those terms permit modification of the work for the customer’s own use and reverse engineering. When distributing the executable in its object code, you have a choice: to either distribute the Library together, provided the source code of the Library is made available in ways similar to the distribution of GPL-ed programs; or to not distribute the Library together, but only use a suitable, shared library mechanism to link with the Library (section 6).

By creating such a category, the LGPL opens a way for LGPL-ed libraries to be used in proprietary programs.

The full text of the LGPL is available at <http://www.gnu.org/licenses/lgpl.html>

12.6. Berkeley Software Distribution (BSD)-style Licenses

The Berkeley Software Distribution (BSD) License was first used for the Berkeley Software Distribution, a version of UNIX developed at the University of California at Berkeley.³⁵ It is easy to follow the BSD license template to create your own license by changing the values of the owner, organization and year appearing in the copyright notice and in the license. Unlike the copyleft licenses, the BSD-style licenses are relatively simple, and have only limited restrictions regarding the software’s use.

Users’ Freedom

- Make copies of the program and redistribute the program, either its source code or its binary code. The distributor is not obliged to provide the source code
- Prepare derivative works and distribute the derivative works, either their source code or binary code. You, the author, are free to choose either FOSS or proprietary licenses for the derivative works
- Incorporate the program into proprietary software

The original BSD License (4-clause BSD) has an advertising clause which has now been rescinded. The revised BSD License (3-clause BSD) is very similar to the MIT License, but the MIT License does not have the “no endorsement for derivative works” clause. There is also the 2-clause BSD, which has taken away the endorsement clause and is most similar to the MIT License.³⁶

12.7. Multiple Licensing

It is important to note that a work can be licensed under more than one license. A license is a choice by you, the copyright-holder, regarding what you think of the relationship between your work and its possible users. But there might be more than one kind of user, and more than one possible relationship. You are, therefore, entitled to choose different kinds of licenses to accommodate different situations.

Take OpenOffice as an example. OpenOffice is dual-licensed under the GPL and the Sun Industrial Standards Source License (SISSL). Although OpenOffice clearly states that it encourages its users to use the GPL in order to participate fully in the OpenOffice.org

35 “MIT License Definition”, June 2004, Available from <http://www.bellevuelinux.org/mitlicense.html>, Internet, Accessed 1 July 2004

36 “WikiReader, Free Software and Free Content”, June 2005, Available from http://en.wikipedia.org/upload/a/a9/WikiReader_Free_Software_and_Free_Contents.pdf, Internet, Accessed 8 July 2004

community, the SISSL is provided as an alternative for those developers and companies who are unable to use the GPL.³⁷

12.8. The source code is open; what about documentation?

12.8.1. GNU Free Documentation License (GNU FDL or FDL)

Good documentation and manuals are extremely important for FOSS, but when they are not licensed as free/open in the same way as FOSS programs are, it is hard for people to make good use of the FOSS programs which they are supposed to illustrate.

Although mainly used as a license for software, the GNU GPL itself can also be used on work which is not software, as long as it is clearly defined what the “source code” means when adopting the license.³⁸ The FSF also provides a license especially designed for documentation. The GNU Free Documentation License (GNU FDL or FDL) is a form of copyleft license for use on a manual, textbook or other document to assure users about the effective freedom to copy and redistribute it, with or without modifications, either commercially or non-commercially.³⁹

By applying the GNU FDL to a document, similar to applying the GPL to a program, you, the author, are granting users the right to make verbatim copies of your work. Since it is a copyleft license, it requires that the copying and distribution of the modification of the FDL-ed work also be licensed under the FDL. But unlike the GPL for software, the FDL has special requirements for making large quantities of verbatim copies.

12.8.2. Creative Commons Licenses

Inspired by FOSS development, the Creative Commons Cooperation advocates the openness of digital content. It urges a more reasonable and flexible layer of copyright in the face of increasingly restrictive default rules.⁴⁰

In 2002, the first version of the Creative Commons Public Licenses (CC licenses) was released. Eleven different licenses identified major concerns which authors consider important. These include:

- Attribution (BY): For any reuse and distribution, it is required that credit is given to the original author.
- Non Commercial (NC): The work cannot be used for commercial purposes.
- No Derivative Works (ND): The work cannot be altered, transformed; derivative works built upon this work are not permitted.
- Share Alike (SA): It is allowed to alter and transform the work, and to prepare derivative works upon this work, as long as the resulting work is licensed to a license identical to this one.

Each represents a unique way of combining these four attributions. You, the author, are free to choose from amongst the eleven licenses, and to decide which one suits your work's requirements.

In 2004, Creative Commons released its second version of CC licenses. Since the requirement of attribution has been greatly adopted by users of CC licenses, the attribution requirement has become the default, and there are only six CC licenses in the second version. The eleven licenses in the first version, however, have not been superseded, and are still available.⁴¹

³⁷ “Licenses”, Aug 2002, Available from www.openoffice.org, Internet, Accessed 28 June 2004

³⁸ Available from www.gnu.org/licenses/gpl-faq.html#GPLOtherThanSoftware, Internet, Accessed 4 Aug 2004

³⁹ Available from www.gnu.org/licenses/licenses.html#TOCFDL, Internet, Accessed 4 Aug 2004

⁴⁰ Creative Commons, “Some Rights Reserved”, Building a Layer of Reasonable Copyright”, Available from <http://creativecommons.org/learn/aboutus/>, Internet, Accessed on 4 Aug 2004

⁴¹ Creative Commons Public Licenses are available from <http://creativecommons.org/licenses/>, Internet, Accessed 5 Aug 2004

The CC licenses have been designed for all kinds of digital content except software projects, including art work, photographs, music and literary text. It does not deal with the open source issue, since all the sources in such work are transparent, and are not compiled into forms which cannot be perceived. Some of the CC licenses do not allow modification and might not be regarded as not “free”. The CC licenses are successful, however, in spreading the idea of freedom and openness among the larger public, which might be unfamiliar with software development.

	Authors' credit is required	Allow commercial use	Allow derivative works	Derivative works should be licensed under the same license as the original work is
CC BY	Yes	Yes	Yes	No
CC BY-NC	Yes	No	Yes	No
CC BY-NC-ND	Yes	No	No	
CC BY-NC-SA	Yes	No	Yes	Yes
CC BY-ND	Yes	Yes	No	
CC By-SA	Yes	Yes	Yes	Yes
CC NC*	No	No	Yes	No
CC NC-ND*	No	No	No	
CC NC-SA*	No	No	Yes	Yes
CC ND*	No	Yes	No	No
CC SA*	No	Yes	Yes	Yes
GNU FDL	Yes	Yes	Yes	Yes

13. Scenarios

For different stakeholders, the use of FOSS might be different from each other. A developer's use of a program might be more labor-intensive than an end-user's use; a software developer's activities might be subject to more restrictions than an end-user's mere running of a program. The following section tries to provide some scenarios as examples to explain the different legal issues which might arise in different uses of FOSS.

13.1. End-user (individual/business/government)

Abul is a public high school teacher. His school can't afford the expensive license fee of proprietary office applications. Although proprietary software companies provide special programs for these schools, Abul still wants to seek an alternative solution to lessen the students' dependence on proprietary software. His friend, Nazlee, is a programmer interested in FOSS development, and she introduces him to a FOSS office application. He and his colleagues download the FOSS office solutions, and teach the students both kinds of applications. Abul is satisfied with the program's performance and introduces it to his colleagues. Gradually, the school's administrative body also begins to use the FOSS solution for various administrative work.

In this case, neither Abul (an individual) nor his school (a public government body) made any modification to the solution they downloaded from the web. They are simply end users.

The situation for end-users is relatively simple. The end-user of a software program might be an individual, a government body, or a business entity. These individual persons or legal entities might have different reasons to use FOSS. Some might be trying to find a cheaper solution or an alternative solution that better suits their needs. Others might wish to use FOSS for better customization. Many more might wish to lessen their dependency on proprietary companies.

13.1.1. Legal Issues Involved

The way you use FOSS solutions might not be very different from the way in which you use proprietary solutions. You download a copy of a FOSS solution or purchase the copy (usually in exchange for some support and services), install it in the computer (by making a copy on the hard disk again), run the program, and have its functions serve your needs.

Your rights concerned here are the right to make copies of the program and the right to run it. (The act of running a program might also be taken as an act of making a copy, but is stated differently in some FOSS licenses. The GPL, for example, has no restrictions on the running of the GPL-ed program, but does regulate the act of copying it). Such rights are granted by all FOSS licenses, and there are fewer legal disputes concerned in the case of the end-user. You might, however, find that some other issues need to be taken into consideration.

Other Considerations - Technical Support

Since you, as an end-user, might not be a computer whiz; even when there are existing FOSS solutions, you might be concerned about the technical support of the FOSS solution that you need. Instead of simply downloading a copy of the FOSS solution which saves you some money, you might choose to purchase a box of FOSS solutions in a shop where a proprietary solution is also available, and sometimes at approximately the price of the proprietary solution. But the difference is that when purchasing Red Hat Linux in the store, or example, you are not paying for the license fee, but for the service and for the support. When the term of service expires, you can choose to pay for another term of service, or ask other available service providers for similar services.

Other Considerations - Customization

When existing FOSS solutions do not fit your needs, you might need to ask individual developers or vendors to customize one for you. In such cases, you, the end-user, or private or public bodies, might want to have written clauses in the contract to have the vendor, or the developer, take complete responsibility for any possible copyright infringement, and compensate you for any possible loss caused by the allegation of infringement. If you are the buyer, you are free to add these clauses in the contract.

Other Considerations - Government Procurement

In cases of government procurement, since FOSS licenses provide different models to the copyright law, the government should be particularly aware when it opens a bid for software solutions or signs a contract with vendors, Existing government bid and contract templates might be drafted under the model of traditional copyright law, and have to be examined, and even revised, if they fail to treat FOSS and proprietary software equally.

13.2. Developer (individual, business)

As a developer, (both the individual developer and business entities) you need to be more careful with the terms and conditions of different licenses because of your intensive use of FOSS. You might not just run and copy the software, but also create derivative works of the software, and distribute these derivative works together with the original program.

As a developer, therefore, in order to contribute to the development of a certain FOSS program, your right to run, to make copies of, to distribute, and to prepare derivative works of the program, are all very essential.

The above rights are granted by all FOSS licenses, for these essential rights are considered important both in the Free Software Definition and in the Open Source Definition. Nevertheless, different FOSS licenses might have different restrictions regarding the practices of such rights, especially about creating and developing derivative works. You should pay particular attention to this, and consult your lawyers with regard to your specific situation as and when needed. Different options might be considered when you participate in different stages of software development.

13.2.1. When starting a new project

Abul's colleague Jolly is the school librarian. The school library is not that big, and it is also open to the villagers. In order to keep accurate records of the books, Jolly seeks her friend's help to write a program for her.

Legal Issues Involved - Choose a License of One's Own

Developer: What does this project mean to me and to others? How do I want others to be involved? What do FOSS licenses say? What are the differences between FOSS licenses?

The situation is relatively simple when you, a developer, are starting your new project without using any existing modules, since you will not have to look through the licenses of any existing modules which you might be interested in using.

Starting a new project all over will not be an easy task either, but if that is what you choose to do, then you might be interested in finding a license that matches your needs. The different characteristics of the FOSS licenses will significantly influence the project's possible development route. Define your requirements before choosing your license.

If, for example, you are a strong follower of free software, you might stick to the GPL or to the LGPL. If you think that you do not need to restrict people's use of your program, you might think that a BSD-style license will suffice. Or when you think that it is better if the development can be controlled in a firm and central line, you might not be interested in BSD-style licenses. On the contrary, if forking is preferred in the future development, the BSD-style license might be a better choice, although there might be difficulties in merging the forked versions back if they are licensed under incompatible FOSS licenses.

Developer: Can I change my mind after licensing my project?

Although you, being the copyright-owner of a project, can always decide to choose another license for your program when previous versions have been already licensed under a certain FOSS license, the rights of the recipients of the previous versions will not be affected since the license grants are irrevocable. The situation will become more complicated if contributions from the community have been incorporated into your newer version, because in that case the copyright-holder might also include contributors.

Developer: I don't like any of the existing FOSS licenses. Can I start a new one?

Although there are already many FOSS licenses, it is still possible that you, the developer, find that you do not like any of them. You are entitled to choose any license for your own project, including a new one that you might have drafted yourself. Creating a new FOSS license, however, requires particular legal knowledge and skill in order to avoid vagueness and loopholes. Also, there are already many FOSS licenses, and it already costs too much in transaction costs to understand them. Creating a new license is not recommended unless you cannot find a suitable one, and have strong reasons for doing so.

13.2.2. When modifying an existing module

The office application that Abul and his school are using is in the English language, and does not support their local language. Although for high school students, using an English interface might not be too much of a problem, it becomes difficult when Abul tries to teach the villagers. Abul consults Nazlee, who has been contributing regularly to FOSS programs, and is also quite familiar with the source code of the office application. She discusses the matter with some of her friends, and, as a team, they begin to localize the application.

Legal Issues Involved - Find out what is the license of the program under modification

When you try to modify an existing module, and when such modification is not solely for your own use, but to distribute it further, e.g. when localizing a program, you need to first find out the module's license. Then you will know what rights you are granted in the license, and what is required when exercising those rights.

Developers: Under the license, what are the rights I am granted, and what are the restrictions in exercising those rights?

With regard to distributing a FOSS work, some FOSS licenses (GPL, LGPL) might require that you, as the distributor, provide both the object code and the source code, or at least provide information regarding the access to the source code.

With regard to modifying a FOSS work, some FOSS licenses (GPL, LGPL, BSD) may require you, the modifier, to provide documentation about the changes you made.

With regard to distributing the derivative works, copyleft licenses require them to be licensed under the same license as the original work, while other FOSS licenses allow you to choose a different license (BSD, MIT).⁴²

In Nazlee's case, they are trying to localize, for example, the dual-licensed OpenOffice. If, as OpenOffice.org suggests, Nazlee and her friends decide to use the GPL, then, as a result, the localized OpenOffice will also be GPL-ed.

Some FOSS licenses, e.g. the MIT License, might allow users to sub-license the original work. This means that when distributing the verbatim copy of the original work, within the scope granted by the original copyright-holder, the distributor may choose a different license and become a licensor himself or herself. In such cases, when you create a derivative work and distribute it together with the original work, you can choose to become a licensor of both the works, and simplify the legal relations between the two parties. If a sub-license is not allowed, the licensor of the original work will be the original licensor, while the licensor of the derivative work will be you, who prepared the derivative work.

13.2.3. When integrating different FOSS modules into one service

Nazlee works in AA Software Inc. To better oversee the many different projects they are developing, the organization has developed a project management system by integrating different FOSS modules. The management system is for internal use only, but since it is extremely useful, they also plan to make its commercial distribution in the future.

This might be a relatively complicated situation; integrating the adopted modules together also creates derivative works of these modules.

Legal Issues Involved - Find out what are the licenses of the programs being integrated, and study at the compatibility between these licenses

In this case, it is essential that AA finds out the licenses of each module. If they happen to be licensed under the same license, e.g. the GPL, then the process will be relatively simple. Since no matter how these modules are linked with each other, no matter how they might need to modify the modules to integrate them together, the integrated system will still be licensed under the GPL. The situation is similar when all modules are licensed under the BSD License, but in this case, AA might be able to choose a proprietary license for the modified modules and the integrated system.

But if some of the modules are licensed under the GPL, and some of the modules are not, then AA will have to look at the compatibility between the different licenses. When two licenses are compatible, the two modules licensed under the two licenses can be combined into a larger work while complying with both licenses.⁴³ The FSF provides a list of GPL-compatible and GPL-incompatible FOSS licenses.⁴⁴

When combining a GPL-ed program and a BSD-ed (GPL-compatible) into a larger program, the larger program will be GPL-ed to meet both the requirements of the GPL-ed program and the BSD-ed program. When some of the modules are GPL-ed but some of the modules are GPL-incompatible, integrating them into a larger system might mean more than the mere aggregation of the modules. AA must then decide which module is more important, and replace the other one with a module that uses a compatible license.

The licenses used in different modules and the way they are combined together will decide how the integrated system can be licensed and distributed.

42 The FOSS licenses are many and they might differ with one another in different ways. "A Comparison of Open Source Licenses" provides an analysis of 15 commonly used FOSS license, Available from <http://www.openfoundry.org/en/archives/000388.html>, Accessed 5 July 2004.

43 "What does it mean to say that two licenses are compatible?" Available from <http://www.fsf.org/licenses/gpl-faq.html#WhatIsCompatible>, Accessed 7 July 2004; "FAQ on Open Source Licenses", Available from <http://www.openfoundry.org/en/archives/FAOonOSL.pdf>, Accessed 7 July 2004

44 "Various License and Comments About Them", Available from <http://www.fsf.org/licenses/license-list.html#GPLCompatibleLicenses>, Accessed 7 July 2004

Other considerations - choice of law and choice of venue clauses

Finally, for you, the developer, who is able to choose licenses for your programs, whether you have started your own programs, or are allowed to choose licenses for the derivative works that you have prepared, beware that many of the OSI-approved licenses have been derived from proprietary software companies. Some of these licenses have been designed to meet their company's policy and strategy, and might not be a good choice for you if you are a general developer. Some technical issues, e.g. the choice of law and the choice of venue clauses (which can be found in the Qt Public License, the Mozilla Public License, and the Common Public License, etc.), might be significant in a real lawsuit, and need to be taken into consideration.

13.3. Vendor/Producer (Business)

Nazlee and her friends have made the localized version of the FOSS office version available for use. AA Software Inc. is very interested in this application and it has also developed some other small but useful programs to facilitate administrative work. They package the localized office application together with their own programs (licensed under their proprietary license). The package sells successfully. A few months later, AA also decides to make a commercial distribution of the project management system which they have integrated from different FOSS modules.

13.3.1. Simple Distribution

In the former situation, both FOSS programs and proprietary programs are being distributed in one package. For the FOSS application, they are merely distributors, and they have to distribute it as its FOSS license requires. For the proprietary programs, AA holds the copyright, and is able to decide about the license and the ways of distribution. It is acceptable to put FOSS applications and proprietary applications into one distribution, e.g. on one CD-ROM, if the applications function separately and do not link together to create any derivative work.

13.3.2. Distribution of Integrated Systems

In the case of an integrated system distribution, it is up to the licenses of the differently integrated modules and the ways in which they are combined. As explained above, AA needs to first make sure that the licenses of the different modules allow AA to combine them together. These licenses will also suggest the ways in which AA can distribute the integrated system.

FOSS is also used in embedded systems. Many devices, e.g. cell phones, handhelds, digital cameras, and DVD players, now use FOSS-embedded systems. The device manufacturers use FOSS to lower their costs when developing new products, but not to distribute FOSS itself.

13.3.3. Government-sponsored Projects

The FOSS movement and rapid FOSS development have been receiving much attention from the FOSS community, academics and policy-makers alike. In some Asian countries, governments work with PC manufacturers and vendors to provide affordable PCs bundled with FOSS operating systems and office applications.^{45,46} These efforts have successfully lowered the prices of owning a PC, and made proprietary software giant Microsoft reduce its prices.^{47,48} Governments also back FOSS, generate FOSS-related projects, and promote FOSS as a national technology policy.⁴⁹ Some government academic institutes have started to work on

45 "Malaysian 'People's PC'- Microsoft experience "Thailand Linux" pain all over again", Mar 2004, Available from http://www.asiaosc.org/article_191.html, Accessed on 7 July 2004.

46 Koanantakool, Thaweesak, "A Case for Nation-wide PC Distribution", Nov 2003, Available from <http://www.asia-oss.org/>, Accessed 7 July 2004

47 Chai, Winston, "Microsoft cuts prices in Thailand", June 2003, Available from http://news.com.com/2100-1012_3-1019067.html, Accessed on 7 July 2004

48 Chai, Winston, "Microsoft Cuts Prices for Malaysia", Mar 2004, Available from http://news.com.com/Microsoft+cuts+prices+for+Malaysia/2100-1016_3-5168029.html?tag=st.rc.targ_mb, Accessed on 7 July 2004

49 Related links available from <http://uwstudent.org/wiki/OpenSourceInGovernment>, Accessed 8 July 2004; and also available from http://www.asiaosc.org/enwiki/page/Ideas_for_OSS_policy.html,

FOSS-related projects even before their governments begin to notice the potential of FOSS and formulate policies about their position regarding it.

On the FSF-maintained FAQs list about the GPL, there are also questions about whether the US Government can release a program under the GPL, or release improvements to a GPL-ed program.⁵⁰ Situations might differ from country to country, and from case to case, under different government regulations in different countries. Most government regulations regarding government-sponsored projects are drafted under the current structure of national and international intellectual property law, and might be more national, economically protective, and unfamiliar, or even unfriendly to FOSS licensing and developing models.

The following are two cases of government-funded FOSS studies. The first one is about FOSS-related studies made in a government research institute without a related government policy. The second one is about a national FOSS project itself.

Government-funded FOSS Projects - Cases from the Asia-Pacific Region

FOSS Project under a Government-affiliated Research Institute – Multi-lingual Editor, Japan⁵¹

Emacs is a multi-lingual text editor first developed by Richard Stallman in MIT. After the GNU project started in 1984, the development of GNU Emacs began, and was first released in 1985. The GNU Emacs was released under the GPL.

The Japanese governmental research institute, Electrotechnical Laboratory (ETL) started to work on the multi-lingual information processing and integration of GNU Emacs and Mule (a multi-lingual text editor based on Emacs and later merged into GNU Emacs as MULE) in the mid 1990s, but there were various issues regarding copyright. The developer ETL was a government research institute, and the licensing model in the GPL was very different from copyright law, so no one was able to decide whether ETL could assign the code to the FSF and release the code under the GPL. As a result, ETL never officially release the code, but expediently released the trial versions. Negotiations between ETL and the FSF resulted in a special agreement. The FSF agreed not to require ETL to assign the copyright of the modified code to the FSF, and ETL agreed to grant the FSF the right to use the code. This was for the first time that part of the code in Emacs did not belong to the FSF.

In 2001, ETL was reorganized into the National Institute of Advanced Industrial Science and Technology (AIST). Although AIST is still a government-funded institute, it is an independent organization, and AIST's assets are not national properties. It seemed then that AIST would be able to release the code under the GPL. It took another year of internal negotiations for AIST to decide that it is entitled to release its works and choose the licenses of its works as well. It was difficult to convince AIST about the advantages of adopting the GPL. According to Dr. Kenichi Handa, a senior researcher at AIST, the main reasons behind the final decision still remain a mystery.

This happened before the Japanese Government formed a clear position regarding FOSS development. In an open source conference between Asian countries, where Dr. Handa was invited to discuss the development of Emacs, Shuichi Tashiro, the leader of the Japanese FOSS project under the Ministry of Economics, Trade and Industry, added that the Japanese Government has made the necessary regulatory revisions to give the developer of government-funded projects the copyright (and thus the right to choose the license), as long as the law is applied from the beginning of the project.

National FOSS Project – Taiwan

Under pressure from the Congress, the Taiwanese Government started planning a national FOSS project in 2002, and began to allocate budgets for 5-year FOSS projects. The Ministry of Economics (MoE) was assigned to structure, sponsor and oversee the sub-projects.

Accessed 8 July 2004

50 Available from <http://www.fsf.org/licenses/gpl-faq.html#GPLUSGov> and <http://www.fsf.org/licenses/gpl-faq.html#GPLUSGovAdd>, Accessed on 10 July

51 Handa, Kenichi, "Development of Multi-Lingual Editor", 2003, Available from <http://www.asia-oss.org/nov2003/present/handa/handa.html>, Accessed on 10 July

Under general government regulations, even the results could be copyrighted by the entity that carried out the government-funded projects, and the applications of the results of projects were subject to certain principles. So, unless it would be more beneficial for the (national) development of science and technology, the results had to be:

- Licensed for a fee
- Licensed to Taiwanese institutes or firms
- Used or manufactured within Taiwanese jurisdiction

Even though exceptions might be made for FOSS projects, while national FOSS development was assigned to the MoE, which bears a more important task in protecting national interest and national economic competency, the more protective and restrictive regulations were applied to national FOSS projects. Under the MoE regulations, only the third principle (used or manufactured within Taiwanese jurisdiction) could be made as an exception. Such principles were not in line with the FOSS development model, and thus made it difficult for sub-projects under the general FOSS project to release the code.

Since the task of FOSS projects would be hindered by such regulations, questions regarding this issue were raised in the first year of the 5-year FOSS project. Different government bodies related to FOSS projects met up several times to find a solution, but because the FOSS licensing model was alien to the models they were used to, the problem remained unresolved, and the code developed in the first year was unable to be officially released.

The most recent negotiations were held in May 2004, where different government bodies came to a conclusion. The MoE would submit a case to the Administrative Yuan (the highest administrative body) regarding applying the general rules to FOSS projects, seeking an official interpretation from the Government that FOSS meets the exception clause and is exempt from the principles. Meanwhile, the MoE would be looking at the possibility of revising the MoE regulations. And while some code-generating FOSS projects have been assigned to the National Science Council and are applying the general rules, hopefully the FOSS projects will be able to release the code under FOSS licenses in the near future.

Online Legal Resources and Materials

- A Comparison of Open Source Licenses, <http://www.openfoundry.org/en/archives/000388.html>
- Apache License and Distribution FAQ, <http://www.apache.org/foundation/licence-FAQ.html>
- Copyrights and Software Protections by Patents and Copyrights, <http://www.ladas.com/Patents/SoftwareProtectionIndex.html>
- Electric Frontier Foundation, <http://www.eff.org/>
- FAQ on Open Source Licenses, <http://www.openfoundry.org/en/archives/FAQonOSL.pdf>
- Feature, Open Source Families and Facts, <http://www.unixreview.com/documents/s=8925/ur0312b/>
- FLOSS Concept Booklet, http://wikibooks.org/wiki/FLOSS_Concept_Booklet
- Foundation for a Free Information Infrastructure, <http://www.ffii.org/>
- Free Software Foundation, <http://fsf.org>
- Frequently Asked Questions about GNU GPL, <http://www.gnu.org/licenses/gpl-faq.html>
- Groklaw, <http://www.groklaw.net/index.php>
- History of Copyright - A Chronology, <http://www.musicjournal.org/01copyright.html>
- Infochange - Intellectual Property Rights, http://www.infochangeindia.org/Intellectual_Property_Rights.jsp
- IP Justice, Campaign for an Open Digital Environment, <http://ipjustice.org/CODE/>
- Journal of Information, Law and Technology, <http://elj.warwick.ac.uk/Jilt/>
- League for Programming Freedom, <http://lpf.ai.mit.edu/>
- Mozilla Relicensing FAQ, <http://www.mozilla.org/MPL/relicensing-faq.html>
- Netscape Public License FAQ, <http://www.mozilla.org/MPL/FAQ.html>
- Open Source Initiative, <http://opensource.org/>
- Open Source License Law Resource Center, <http://www.denniskennedy.com/opensourcelaw.htm>
- Open Source Licensing, <http://www.anu.edu.au/people/Roger.Clarke/EC/OSLic.html>
- Quiz to Test Your Knowledge of the GPL and LGPL, <http://www.gnu.org/cgi-bin/license->

- [quiz.cgi](http://en.wikipedia.org/upload/a/a9/WikiReader_Free_Software_and_Free_Contents.pdf)
WikiReader, Free Software and Free Content,
http://en.wikipedia.org/upload/a/a9/WikiReader_Free_Software_and_Free_Contents.pdf

14. Glossary

Copyleft

Proposed by free software advocates, copyleft is an alternative idea to the present image of human creative activities defined in copyright law. The copyright law usually confers exclusive rights to copyright holders and limits all others' access to the work. Authors might want to "copyleft" their works to grant certain rights to people who are interested in using their works, provided these people also "copyleft" all the works they have created based on these works. Although copyright and copyleft might represent very different ideas about the relationship between authors and their works, copyleft is not against copyright law. On the contrary, without the rights granted by the copyright law, authors will not have the power to copyleft their works.

Copyright

A set of exclusive legal rights granted by governments to an author regarding the use of an original expression (including literary works, movies, music compositions, paintings, and software, etc.) granted exclusively to the holder. Copyright is applied to a work upon its creation. Except for the limitation set by copyright law, any use of a work without the copyright-holder's consent is regarded as an illegal infringement. Please note that copyright law protects only the particular way of expressing the ideas, not the ideas themselves.

Copyright-holder

An individual or legal entity entitled to exclusive rights under copyright law. It is usually said that copyright law is to protect authors of creative works, but most of the rights protected are treated as property rights and can be transferred. Many copyright-holders are not authors of the works themselves but their employers.

Derivative works

Copyright law is applied to every work once it is created. When agreed by the copyright-holder, anyone can create derivative works based on this (original) work. A newer version of a program, for example, might contain all or part of the code in its earlier version, thus the newer version is a derivative work of the earlier version. The translation of a document is also regarded as a form of derivative works.

Distribution/Redistribution

The distribution of the copies of a work is also an exclusive right granted to the copyright-holder. In FOSS licenses, all receivers of copies of a program are allowed to make further distributions. The term "redistribution" may be used when emphasizing that the distributor has received the program from somewhere and is distributing it further.

Fair Use

Copyright law is about a balance between private and public interests. "Fair use" is developed to limit copyright protections and to provide greater access to the works to the general public.

When a work is used without seeking consent from the copyright-holder for purposes of comments, criticism, news reporting, research, scholarship or teaching, it might not be considered as an infringement. The following factors may be considered by the court in deciding whether a case falls in fair use or is an infringement of copyright:

- The purpose and character of the use, including whether such use is of a commercial nature or is for non-profit educational purposes
- The nature of the copyrighted work
- The amount and substantiality of the portion used in relation to the copyrighted work as a whole
- The effect of the use upon the potential market for or value of the copyrighted work

License

Copyright or patent holders usually require users to accept the terms and conditions of a license as a pre-requisite to allow their use of the copyrighted works.

Multiple Licensing

The copyright-holder of a work can have various ways to make use of his or her work. The terms and conditions they would want users to accept can differ from case to case. For example, the copyright-holder of an “editor” might be willing to issue an academic license, which is cheaper and more affordable for students, while commercial licenses are adopted when selling the program to commercial entities. A copyright-holder can also decide to license a work under both FOSS licenses and proprietary licenses in order to achieve different purposes.

Public Domain

The public domain is a pool that consists of creative works that are not protected by copyright law and can be used freely. Works falling into this category might be cultural heritage that came into existence before copyright law; or works that were once protected by copyright law, but their terms of protection have expired; or works that its copyright-holder decides not to claim copyright to. In the latter case, such disclaimers must be explicitly made.

Source Code

Source code is written in a special kind of language designed for programming. A program in its source code form might not be easy for lay-persons to understand, but it is intelligible to trained programmers. When the source code is converted to machine-readable form, programmers have difficulties understanding and modifying the program. Access to the source code is, therefore, a pre-requisite for the development of FOSS, and a principle embraced by all FOSS licenses.

Details regarding “source code” can be found in the “Glossary” section of an introductory primer, *Free/Open Source Software, A General Introduction*, available from http://www.iosn.net/downloads/foss_primer_current.pdf

Sub-licensing

When copyright-holders license their work to someone else, they can also choose to let the licensee sublicense their work, i.e., when the licensee distributes the work, within the scope of rights granted by the licensor, the licensee is not only a (re)distributor, but can also become a licensor of a sub-license between them and the other party (licensee of a sub-license).

Most FOSS licenses, however, do not grant people the right to sub-license. For example, A is the copyright holder of X program. B receives a copy of X and distributes more copies. C receives the copy from B. If A does not grant B the right to sublicense, both B and C receive the license directly from A. If A grants the right to sublicense program X, within the scope of the rights granted by A, B may start a new license, and himself become a (sub)licensor of program X.

Warranty Disclaimer

A warranty is a guarantee of the liability of a product. A warranty disclaimer can always be found in all FOSS licenses. Such clauses are designed to protect the author of FOSS programs, for these programs are licensed without royalty and changes might be added in its development. Though the FOSS programs themselves are royalty-free and disclaim warranty, nevertheless, vendors of FOSS programs can always provide its customers with a warranty and various kinds of support for a fee.

- i Wheeler, David, "Why OSS/FS? Look at the Numbers!" [**Home** page online]; available from http://www.dwheeler.com/oss_fs_why.html; Internet; accessed on November 7, 2003.
- ii "The Free Software **Definition**" [**home** page online]; available from <http://www.fsf.org/philosophy/free-sw.html>; Internet; accessed on November 9, 2003.
- iii Open Source Initiative [home page online]; available from <http://www.opensource.org>; Internet; accessed November 8, 2003.
- iv Raymond, Eric S., "The Cathedral and the Bazaar" [home page online]; available from <http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>; Internet; accessed on November 7, 2003.
- v Raymond, Eric S., "The Cathedral and the Bazaar" [home page online]; available from <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html>; Internet; accessed on November 7, 2003.
- vi Bengtsson, Lassing, Bosch, van Vliet, "Analyzing Software Architectures for Modifiability"; available from <http://www.cs.rug.nl/~bosch/papers/SAAModifiability.pdf>; Internet; accessed on November 7, 2003.
- vii "A Brief History of Free/Open Source Software Movement" [home page online]; [available from http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html](http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html); Internet; accessed on November 7, 2003.
- viii "An Open Letter **to** Hobbyists by Bill Gates – 1976"; available from http://www.tranquileye.com/cyber/1976/gates_open_letter_to_hobbyists.html; Internet; accessed on November 7, 2003.
- ix Moody, Glyn, "Rebel Code", Penguin Books, London, England, 2001.
- x "A Brief History of Free/Open Source Software Movement" [home page online]; available **from** <http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html>; Internet; accessed on November 7, 2003.
- xi "History of the OSI" [home page online]; available from <http://www.opensource.org/docs/history.php>; Internet; accessed on November 7, 2003.
- xii Scannell, Ed. "Linux takes the operating system scene by storm", *Infoworld.com*; available from http://archive.infoworld.com/supplements/99poy_drv/99poy_linux.html; Internet; accessed on November 7, 2003.
- xiii Leonard, Andrew, "An Alternative Voice: How the Tech-Poor Can Still Be Software Rich", 28 June 2001, The International Herald Tribune Online; available from <http://www.iht.com/cgi-bin/generic.cgi?template=articleprint.tmplh&ArticleId=24330>; Internet; accessed on November 7, 2003.
- xiv Pescatore, John, "Commentary: Another worm, more patches", 20 September 2001, *CNet News.com*; available from <http://news.com.com/2009-1001-273288.html?legacy=cnet&tag=nbs>; Internet; accessed on November 8, 2003.
- xv Luening, Eric, "Windows users pay for hacker insurance", 29 May 2001, *CNet News.com* [home page online]; available from <http://news.com.com/2100-1001-258392.html?legacy=cnet>; Internet; accessed on November 8, 2003.
- xvi Ghosh, Krieger, Glott, Robles, "Free/Libre and Open Source Software: Survey and Study. Part 2B: Open Source Software in the Public Sector: Policy within the European Union", June 2002; available from http://www.infonomics.nl/FLOSS/report/FLOSSFinal_2b.pdf; Internet; accessed on November 8, 2003.
- xvii Najani, Niranjan, "Free as in Education", available from <http://www.maaailma.kaapeli.fi/FLOSSReport1.0.html>; Internet; accessed on November 8, 2003.
- xviii Vaughan-Nichols, Steven J., "Can You Trust This **Penguin?**" 1 November, 1999, *ZDNet SmartPartner*. Article no longer available from ZDNet site but archived at <http://web.archive.org/web/20010606035231/http://www.zdnet.com/sp/stories/issue/0.4537.2387282.00.html>; Internet; accessed on November 8, 2003.
- xix Wheeler, David, "Why OSS/FS? Look at the Numbers!" [**Home** page online]; available from http://www.dwheeler.com/oss_fs_why.html; Internet; accessed on November 8, 2003.
- xx "The Web Standards Project: Fighting for Standards in our Browsers" [**home page online**]; available from <http://archive.webstandards.org/upgrade/>; Internet; accessed on November 8, 2003.
- xxi Ghosh, Krieger, Glott, Robles, "Free/Libre and Open Source Software: Survey and Study. Part 2B: Open Source Software in the Public Sector: Policy within the European Union", June 2002; available from http://www.infonomics.nl/FLOSS/report/FLOSSFinal_2b.pdf; Internet; accessed on November 8, 2003.

- ^{xxii} Ghosh, Krieger, Glott, Robles, "Free/Libre and Open Source Software: Survey and Study. Part 2B: Open Source Software in the Public Sector: Policy within the European Union", June 2002; available from http://www.infonomics.nl/FLOSS/report/FLOSSFinal_2b.pdf; Internet; accessed on November 8, 2003.
- ^{xxiii} Ibid.
- ^{xxiv} "Doc1: Halloween Documents" [home page online]; available from <http://www.opensource.org/halloween/halloween1.html>; Internet; accessed on November 8, 2003.
- ^{xxv} **Business Software Alliance-IDC Global Software Piracy Study; available from <http://www.bsa.org/globalstudy/upload/2005-2006%20Global%20Piracy%20Study.pdf>; Internet; accessed on May 25, 2006.**
- ^{xxvi} "Frequently Asked Questions" [home page online]; available from <http://www.lisa.org/info/faqs.html#gil>; Internet; accessed on November 8, 2003.
- ^{xxvii} Walsh, Mary Williams, "Microsoft in War of Words", *Los Angeles Times*; available from http://www.tungutaekni.is/ymis_frodleikur/war_of_words.html; Internet; accessed on November 8, 2003.
- ^{xxviii} DiCarlo, Lisa, "PeopleSoft Jumps On The Linux Train", *Forbes.com*; available from http://www.forbes.com/technology/2003/05/06/cx_ld_0506psft.html; Internet; accessed on November 8, 2003.
- ^{xxix} Herrington, Jack, "Is Documentation Holding Open Source Back?" *DexX.com* [home page online]; available from <http://www.devx.com/devx/editorial/11839>; Internet; accessed on November 8, 2003.