LV 0974 „Spezielle BWL PM A: Vertiefungskurs VI – Wirtschaftsinformatik“

Winter term 2006/07

THE CONCEPT OF FREE SOFTWARE:

Free Software Foundation, GNU and GNU General Public License

Tutor:

ao.Univ.Prof. Dr. Rony G. Flatscher

Author:

Patricia Böhm (0050295)

'free' as in 'free speech', not as in 'free beer'

Richard M. Stallman: Free Software, Free Society:
Selected Essays of Richard M. Stallman.
GNU Press, Boston, MA 2002.

Table of Contents

Table of Figures

1. Introduction

In modern times people in the industrialized world are surrounded by an innumerable number of hi-tech devices, which sometimes almost seem to take control of our lives. As people become increasingly dependent on technical and technological development, both commercial software producers and individual developers are releasing new computer programs day-to-day. As a consequence of the so-called information overload and the ever-growing importance of hi-tech devices, terms like 'freeware', 'open source' or 'free software' are often being used in an inflationary way. People increasingly use these terms without really knowing what they signify or what they were intended to imply when they arose for the first time.

The present paper intends to clear up parts of the confusion of ideas concerning the subject area of free software. First of all, it gives a basic categorization of software and definitions of the relevant terms used in this paper, followed by a detailed overview of the original concept of free software as it was primarily formulated by Richard Stallman, who become the figurehead of the Free Software Movement. Furthermore, the author will oppose the characteristics of free software and open source software by pointing out the different views and visions as well as emphasizing the similarities those two concepts share.

Second, the paper outlines the basic principles of copyright law and contrast them with the concept of copyleft, which was also characterized by Richard Stallman. Subsequent to portraying the history of the Free Software Foundation (FSF), which was particularly established to spread the ideals of the Free Software movement, the organizational profile as well as the principles of the FSF are discussed, leading to an overview of the GNU project.

The development of the GNU General Public License (GPL), headed by Richard Stallman and his advocates, represents another main part of this paper. Starting with a basic categorization of software and licensing models, the author discusses the characteristics of the GNU GPL and the steps of development of the new version, the GPLv3, as well as the reasons for the reissue of the GPL, which over the years has become the dominant

license for free and open source software. In particular, the paper covers the question of Digital Rights Managements (DRM) and the attitude of the Free Software Foundation towards DRM technology which from the FSF's point-of-view places limits on how consumers can play movies, music or other digital content.

Finally, the paper presents two specific groups of stakeholders of free software. It goes into detail about the factors of motivation for developers and programmers, i.e. why they release or further develop free software, giving permission for anyone to use, copy, distribute and even modify the software. In addition, the various advantages of which users of free software benefit as well as the disadvantages from which users may be afflicted are pointed out.

2. Characteristics of Free Software

2.1. Basic categorization of software

In general, software can be categorized as either free or non-free software [cf. Reit04, 87]. On the one hand, free software[1] is software that comes with permission for anyone to use, copy, and distribute, either verbatim or with modifications, either gratis or for a fee. On the other hand, non-free software is any software that is not free, including semi-free software and proprietary software. Semi-free software is software that is not free, but comes with permission for individuals to use, copy, distribute and modify (including the distribution of modified versions) for non-profit purposes [cf. Free05d]. Proprietary software is software offered for sale or license where the vendor controls the source code of the software [cf. Info07]. Its use, redistribution or modification is either prohibited, or requires to ask for permission, or is restricted so much that it can effectively not be done freely [cf. Free05d].

The term 'freeware' has no clear accepted definition, but it is commonly used for software where redistribution is permitted but modification is not. As the source code of freeware is not available, this term should be rigorously distinguished from the term 'free software' [cf. Free05d].

By contrast to freeware, shareware is software which comes with permission to redistribute copies, but demand that anyone who continues to use a copy of the software after a certain so-called trial-period is required to pay a license fee. For most shareware, source code is not available. Thus, the program can not be modified by the users. In addition, shareware does not come with permission to make a copy and install the program without paying a license fee, not even for individuals engaging in non-profit activity. For the reasons mentioned above, it is evident that shareware is not free software, or even semi-free software [cf. Free05d].

Besides the categorization as free or non-free software, a distinction between commercial and non-commercial software can be made. Commercial software is often associated with

1 Richard Stallman used the term 'free software' to distinguish his concept from traditional proprietary software [cf. Möll05, 61].

quality, payment, interest in making a profit and motivation for long-term maintenance [cf. Reit04, 87].

As for the process of development, software development can either be open or closed. In this context, open means that a large number of people have access and can contribute to the source code and the decisions made within a software development project. Generally, free software tends to be developed open, but free software can also be subject to closed development. By contrast, there are certain limits for open development of proprietary software. Nonetheless, some developers of proprietary software try to take advantage of the benefits and the image of open software development, e.g. Microsoft's 'Shared Source' program or Sun's 'Sun Community Source License' [cf. Reit04, 87].

2.2. Free Software Definition

The fundamental interpretation of free software is 'that software can not be owned' [cf. Kard04, 8]. According to the definition of the Free Software Foundation, free software is a matter of the users' freedom to run, copy, distribute, study, change and improve software. More precisely, it refers to four kinds of freedom for the users of free software [cf. Free05a]:

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

A program is free software if users have all of these freedoms. Thus, a user needs to be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Among other things, being free to do these things means not having to ask or pay for permission [cf. Stal02, 41]. To make these four fundamental freedoms effective in practice, users need to have full access to source

code[2] and need to be able to make changes to source code without constraint [cf. Webe04, 48]. Moreover, the freedoms quoted above must be irrevocable as long as they are not being violated, because software is not free if a developer of software has the power to revoke the license [cf. Stal02, 42].

In order to understand the concept of free software, which was mainly characterized by Richard Stallman, the FSF stresses to think of 'free' as in 'free speech', not as in 'free beer' [cf. Free05a]. Since free refers to freedom, not to price, there is no contradiction between selling copies and free software [cf. Webe04, 47]. Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution [cf. Stal02, 42]. In fact, the freedom to sell copies is crucial since it is an important way to raise funds for free software development [cf. Webe04, 48].

Figure 1 illustrates the concept and the fundamental values of free software. It shows how the various components and parties involved in the process of developing and using free software products interact.

2 The FSF states the following: 'The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.' [cf. Free05e].

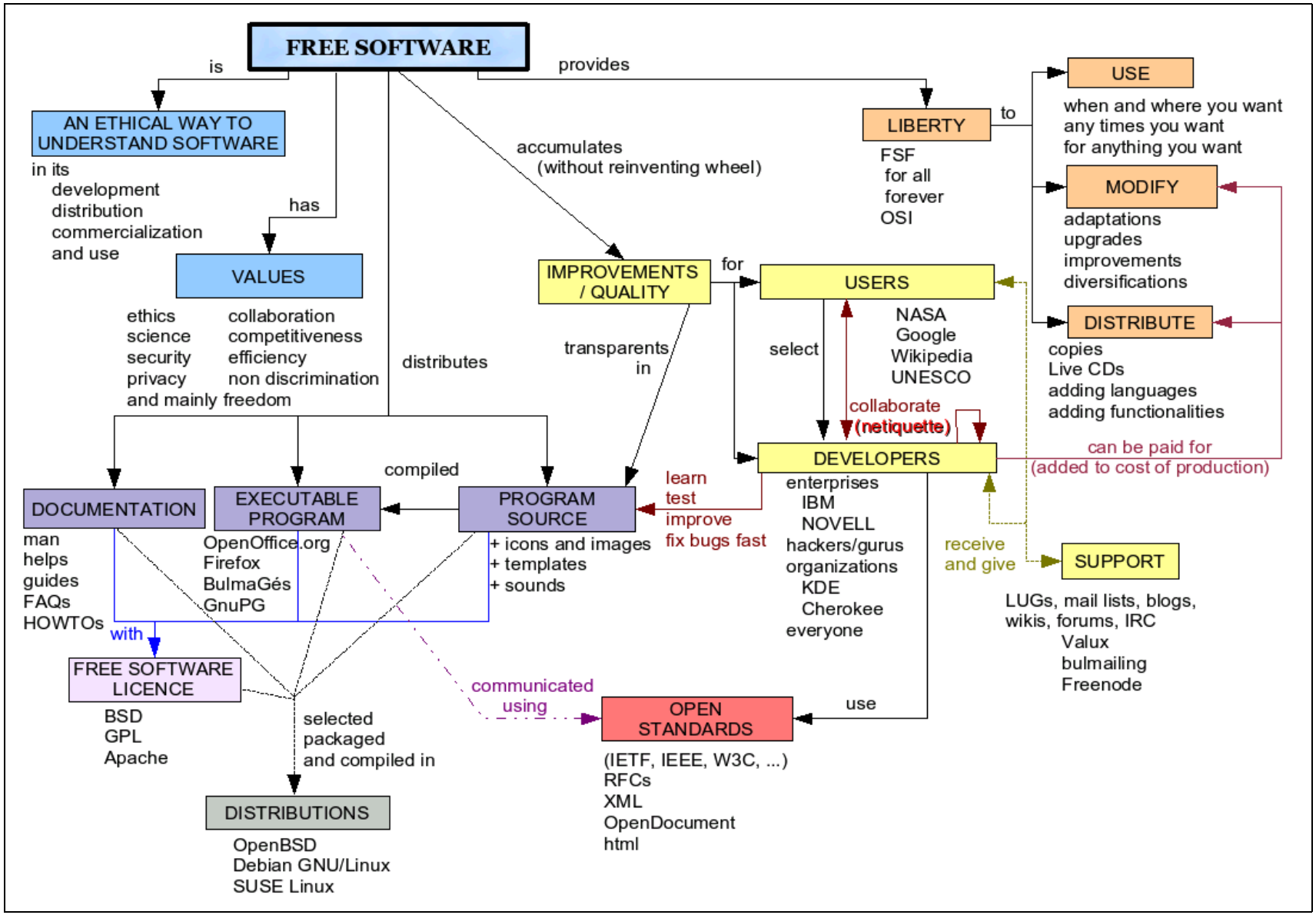

Figure 1: Concept Map of Free Software [cf. GnuE06]

2.3. Free Software versus Open Source Software

The term 'open source' was first suggested at the Initial Conference of the Open Source Initiative (OSI), which was founded by in 1998. As the term 'free' is not only ambiguous but had also become an indecent and controversial term in *The Land of the Free*, it was the manifested aim of the OSI to replace Richard Stallman's term 'free software' with a term that could also be used to pitch non-proprietary software to business executives [cf. Gras04, 230].

Instead of the four fundamental freedoms formulated by Richard Stallman, only the principle of open source is to the fore [cf. Möll05, 62]. According to the Open Source Initiative, the definition of open source software comprises the following requirements [cf. Kard04, 8]:

- Source code must be distributed with the software or otherwise made available for no more than the cost of distribution.
- Anyone may redistribute the software for free, without owing royalties or licensing fee to the author.
- Anyone may modify the software or derive other software from it and then distribute the software under the same terms.

The OSI does not have a position on whether ideas can be owned, whether patents are good or bad, or any of the related controversies. The economic self-interest arguments for open source are considered strong enough that nobody needs to go on any moral crusade about it. The OSI describes itself as a 'marketing program for free software' [cf. Kard04, 9].

One of the basic ideas of the Open Source Initiative was to focus on the technical aspects of free software in order to be able to promote it more easily within the company environment. Therefore, the OSI restricts itself to the treatment of technical aspects and methods of development. The Free Software Foundation goes much further by considering cultural effects and impacts on society as well [cf. Reit04, 85].

The Free Software movement and the Open Source movement are like two political camps within the free software community. The fundamental difference between the two

movements is in their values, their ways of looking at the world [cf. Stal02, 55]. Open source is a development methodology, whereas free software is rather a political philosophy or social movement [cf. Kard04, 6]. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one, and non-free software is considered a suboptimal solution. By contrast, for the Free Software movement, non-free software is a social problem, to which free software is the solution [cf. Stal02, 55].

The terms FOSS (Free and Open Source Software) and FLOSS (Free, Libre and Open Source Software) probably have partly emerged from ignorance. One the other hand, they also represent a half-hearted attempt to politically embrace the two camps within the Free Software movement [cf. Reit04, 85].

3. Copyright Law versus Copyleft

3.1. Basic principles of Copyright Law

Copyright grants authors of protected works a comprehensive set of exclusive rights in order to control the exploitation of their works [cf. Drei96]. Copyright is automatically attached to every novel expression of an idea, whether through text, sounds, or imagery, under the laws of the United States, as well as through the Berne Convention for European Countries and through the WTO Agreement on Trade-Related Aspects of Intellectual Property Rights for members of the World Trade Organization [cf. StLa04, 1].

Works protected by copyright law can not be copied, displayed, or otherwise commercially exploited by any person other than the creator for the life of the copyright [cf. StLa04, 1]. Under Austrian Law – as well as under US Law – the period protected by copyright lasts for the life of the creator plus 70 years for works of literature (including computer programs), musical arts and visual arts [cf. Bund06]. After the expiration of that period of time, the copyright protection on the work expires as the work goes into the 'public domain'. Anyone is then free to commercially exploit such works by selling copies of those works, creating derivative works based upon them, and by distributing or displaying the work publicly [cf. StLa04, 3].

Amongst others, no person other than the creator has the right under copyright law to create so-called 'derivative works'. These are works that depend upon or develop from the original, copyrighted work [cf. StLa04, 1]. In contrast, a 'transformative derivative work' is one that, although based on a copyrighted work, so fundamentally alters it that a new work results. Such a work is considered a new work for copyright purposes and the holder of the copyright of the work from which the 'transformative derivative work' is derived has no rights over it [cf. StLa04, 3].

Copyright law does not protect any particular idea, but rather only the expression of that idea. This limitation to the expression of an idea is the principal distinction between the applications of patent and copyright. Unlike copyright, a valid patent does not protect the expression of an idea but the underlying substance of it. Furthermore, a copyright does not

need to be registered to be legally effective as copyright comes into force when the protected work is created [cf. StLa04, 2].

Copyright serves as an incentive for individual authors to creation and for the dissemination of the works created. As it also contributes to the industrialized nations' GNP, it is being viewed as trade-related and is being discussed within the framework of international competitiveness, securing of full-employment, and thus as a factor of social well-being [cf. Drei96].

3.2. The concept of Copyleft

Copyleft is a general method for making a program free software and requiring all modified and extended versions of this program to be free software as well [cf. Stal02, 89]. Copylefted software is free software, whose distribution terms do not let re-distributors add any additional restrictions when they redistribute or modify the software. Therefore, every copy of the software must be free software, even in case it has been modified [cf. Free05d]. Thus copyleft guarantees that every user has freedom because anyone who redistributes the software – with or without changes – must pass along the freedom to further copy and change it [cf. Free05b].

The simplest way to make a program free is to put it uncopyrighted in the public domain, which allows people to share the program and their improvements. But this approach also allows people to convert the program into proprietary software by making changes and distributing the result as a proprietary product. Those people who receive the program in that modified form do not have the freedom that the original author gave them. As it is the aim of the Free Software Foundation to give all users the freedom to redistribute and change software, instead of putting software in the public domain it is being copylefted [cf. Stal02, 89].

To copyleft a program, the FSF first states that it is copyrighted. Then distribution terms are added, which are a legal instrument that gives everyone the rights to use, modify and redistribute the program's copy or any program derived from it, but only if the distribution terms are unchanged. In this way, the code and the freedoms become legally inseparable [cf. Free05b]. In other words, copyleft uses copyright law, but flips it over to serve the

opposite of its usual purpose. Instead of a means of privatizing software, it becomes a means of keeping software free [cf. Stal98].

Figure 2 shows the symbol for 'public domain', whereas Figure 3 presents the copyleft symbol, which is the copyright symbol turned in the left direction. Unlike the copyright symbol, both the public domain symbol and the copyleft symbol have no legal meaning.

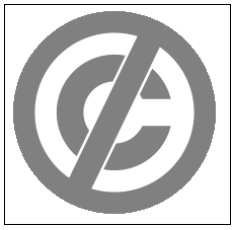

Figure 3: Symbol for public domain [cf. Linu06]

Figure 2: Copyleft symbol [cf. Linu06]

According to the FSF, copyleft provides an incentive for programmers to add to free software, but it also helps programmers who want to contribute improvements to free software get permission to do that [cf. Stal02, 89]. As copyleft is a general concept, it is necessary to use a specific set of distribution terms in order to actually copyleft a program. In actual practice, nearly all copylefted software uses the GNU General Public License [cf. Free05d].

4. The Free Software Foundation (FSF)

4.1. Formation of the Free Software Foundation

In the 1960s and 1970s, the MIT Artificial Intelligence Laboratory in Massachusetts, United States, was a major center for the development of software and particularly computer communications and time-sharing systems. Richard Matthew Stallman (see Figure 4), who was born on March 16, 1953, was one of the programmers working at MIT [cf. Webe04, 46]. He had started his career at MIT in 1971 [cf. Free06e].

Figure 4: Richard Stallman [cf. NoAu06e]

At this time, the MIT was also a place where the intellectual culture was founded on openness, sharing and collaboration. As Richard Stallman once described, the members of the MIT did not call their software 'free software' because 'that term did not yet exist, but that is what it was. Whenever people from another university or company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.' [cf. Webe04, 46].

In the late 1970s and early 1980s, the growth of proprietary software started to show an impact on the MIT community. Many of the best programmers were hired away into lucrative positions in spin-off software firms and the MIT began to demand that its employees sign nondisclosure agreements. In addition, the newest mainframes came with operating systems that did not distribute source code. In fact, researchers had to sign nondisclosure agreements simply to get an executable copy. It was Richard Stallman who led the backlash. According to him, the problem crystallized in 1979 when the MIT

laboratory got a new laser printer from Xerox. As the printer suffered from paper jams, Stallman and his colleagues wanted to deal with this little problem in the same way they had always dealt with problems – by experimenting with and modifying the software so it would work better. When Xerox was not willing to give the source code to the members of the MIT, Richard Stallman was annoyed and frustrated [cf. Webe04, 46 et seq.].

In 1984 Stallman resigned his position at the MIT Artificial Intelligence Laboratory to devote himself to what he called 'free software'. For him, software was not just a tool to run computers. It ultimately was a manifestation of human creativity and expression. Moreover, software represented a key artifact of a community that existed to solve problems together for the common good. For Stallman, proprietary software ran directly against the moral sentiments of a decent society. That's why in 1985 he founded the Free Software Foundation as a non-profit organization to support his work. His goal was to produce an entirely free operating system that anyone could download, use, modify, and distribute freely [cf. Webe04, 47].

4.2. Organization and principles of the Free Software Foundation

The Free Software Foundation is a non-profit organization based in Boston, Massachusetts, United States. It has three major sister organizations around the world:

- FSF Europe (founded in March 2001)[3],
- FSF India (founded in November 2003)[4], and
- FSF Latin America (founded in November 2005)[5].

The logo of the Free Software Foundation is shown below in Figure 5.

Figure 5: Logo of the Free Software Foundation [cf. Fre05a]

3 For further information, see: http://www.germany.fsfeurope.org/
4 http://fsf.org.in/
5 http://www.fsfla.org/

The FSF was founded to spread the ideals of the Free Software movement as well as the use and knowledge of free software [cf. Free06d]. In particular, it was established to support the GNU project (see chapter 5) by assisting administrative, legal, and organizational aspects of the GNU project [cf. Free06f]. Furthermore, the FSF is the principle organizational sponsor of the GNU project. The FSF receives very little funding from corporations or grant-making foundation, but relies on support from individuals [cf. Free06d].

The FSF supports the freedoms of speech, press, and association on the Internet, the right to use and encryption software for private communication, and the right to write software unimpeded by private monopolies [cf. Free06d]. Moreover, the FSF takes free software programs under its wings, offers legal advice and represents the members of the Free Software community in the media [cf. Möll05, 60].

5. The GNU project

GNU is a recursive acronym of 'GNU's Not Unix[6]' [cf. Free06a]. The logo of the GNU project, which exists in several variations, is shown on the right.

Figure 6: The GNU logo [cf. Stal02]

The GNU project was launched in 1984 by Richard M. Stallman to develop a complete UNIX-like operating system which is free software: the GNU system [cf. Free06d]. Being Unix-like, GNU is modular in design. This means that components from third parties can be inserted into GNU [cf. Free06f].

Richard Stallman made the Initial Announcement of the GNU project in September 1983. The GNU Manifesto, which was written by him and has been translated into several other languages, was published in September 1985 [cf. Free06e]. Stallman wrote the GNU Manifesto to ask for participation and support. For the first few years, it was updated in minor ways to account for developments. Over the years, several footnotes were added to help clarify certain common misunderstandings [cf. Stal02, 31].

Bit by bit the several components of the GNU operating systems were put together. But what was still missing was the kernel, the core of the operating system that provides programs access to the system hardware. Since 1990, the programmers working on the GNU system had been striving to complete the operating system with a new kernel named 'Hurd'. But when Linus Benedict Torvalds, a Finnish student of computer science at the University of Helsinki, made his kernel called Linux freely available, it became clear that Hurd would not be completed in the foreseeable future. But Linux might probably have been valueless without the extensive 'GNU toolbox', i.e. the various components of the GNU system that were already available. As Linus Torvalds and Linux became more and more popular, Richard Stallman and the GNU project hardly got appreciation. Stallman and other advocates of the Free Software movement insisted on speaking about

6 Unix was a very popular operating system in the 80s, so Stallman designed GNU to be mostly compatible with Unix so that it would be convenient for people to migrate to GNU. The name acknowledges that GNU learned from Unix's technical design, but also importantly notes that they are unrelated [cf. Free06f].

GNU/Linux instead of Linux, but this term only partly prevailed until today [cf. Möll05, 60 et seq.].

However, the GNU project is not limited to the core operating system, as the FSF states. The Free Software Foundation aims to provide a whole spectrum of software, whatever many users want to have. As a consequence, the GNU project still supports the FSF's mission to preserve, protect and promote the freedom to use, study, copy, modify, and redistribute computer software, and to defend the rights of Free Software users [cf. Free06d].

6. GNU General Public License (GPL)

6.1. Software categories and licensing models

The topic of licensing is one of the most important issues about the free and open-source software industry. The license essentially indicates what companies and developers can and can not do with their software, which code it can or can not be mingled with, and what patent and other protections are afforded to the user [cf. Gall06, D6].

Not only developers but also business people should care about the license and its terms, because those terms define the parameters in which companies can utilize open source software. Both company executives and developers should be aware of what a software license allows and also disallows. A company considering the release of some of its code under open source needs to consider which license best furthers its business goals, whereas developers should be conscious of what license protects a piece of code and what happens when code from several sources, protected under several different licenses, is combined to create a product for resale [cf. Gall06, D6 et seqq.].

Licensing is also important to the future of technology development, as this is one of the principal vehicles by which companies commercialize their developments. Furthermore, it is also how companies share technology and take advantage of the innovations of others. As companies seek to take advantage of open source technologies and platforms and find ways to use open source licensing models to further their business interests, open source-style licensing is continuing to gain in importance [cf. Gall06, D6]. Some companies are moving to a dual-licensing model where software is released under two different licenses. This approach allows users to choose which licensed distribution they want to run [cf. Gall06, D8].

Figure 7 represents the different categories of software and licensing models. Moreover, it points out which licensing models can be used in order to copyleft a computer program. The characteristics of the GNU General Public License are left out at this point; they are elaborated in chapter 6.2.

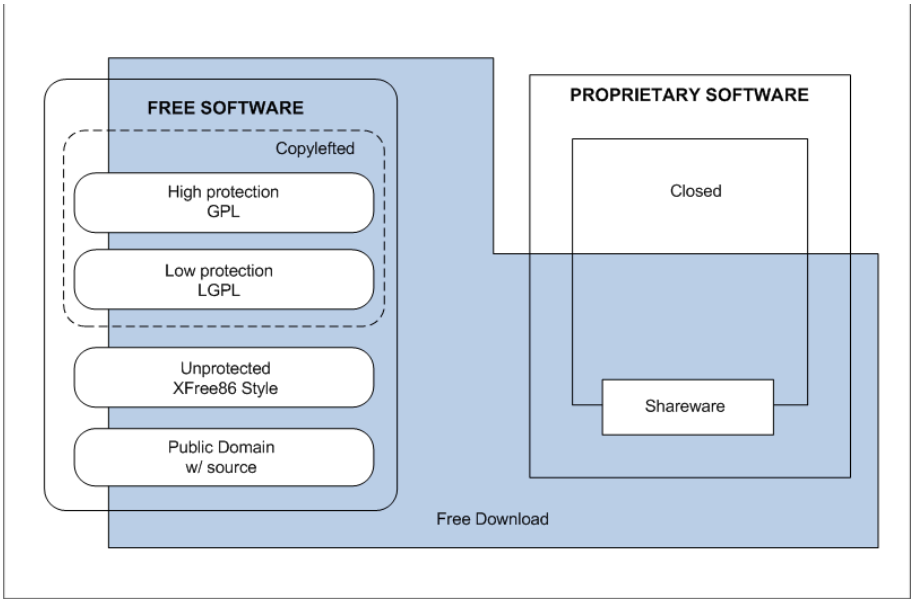

Figure 7: Software categories and licensing models [cf. Free05c]

The GNU Lesser General Public License (GNU LGPL) is a free software license compatible with the GNU GPL. In contrast, it is not a strong copyleft license, because it permits linking with non-free software modules. Between version 2 and version 2.1, the GNU LGPL was renamed from the GNU Library General Public License to the GNU Lesser General Public License to better reflect its actual purpose as it is not just intended for libraries. The FSF recommends the use of the GNU LGPL for special circumstances only [cf. Free06c].

The XFree86 1.1 License is a simple, permissive non-copyleft license, incompatible with the GNU GPL [cf. Free06c]. Under a XFree86 License, software can be distributed without source code and the freedoms of free software, and it can be used as a component for proprietary software [cf. Rei04, 86]. Currently there are several variants of XFree86 and only some of them use this license. Other variants use the X11 license which is compatible with the GNU GPL [cf. Free06c].

Being in the public domain is not a license. Rather, it means the material is not copyrighted and no license is needed. In practice, though, if a work is in the public domain, it might as well have an all-permissive non-copyleft free software license. Public domain status is compatible with the GNU GPL [cf. Free06c].

6.2. Characteristics of the GNU GPL

Richard Stallman reversed the principles of software licensing by developing a license that granted new rights to users instead of taking away rights [cf. Möll05, 61]. The GPL, which is administered by the FSF [cf. NoAu06b] and is based on Richard Stallman's concept of copyleft, uses copyright law to ensure that free software and derivative works from free software remain free. The central idea of the GPL is that it uses copyright law to extend the four freedoms of free software, by preventing any users from adding restrictions that could deny these rights to others [cf. Webe04, 48].

Software that is licensed under the GPL can not be made proprietary. Derivative works from free software must also be free. Furthermore, the GPL does not allow the use of GPL'ed code in any proprietary implementation at all. It is not permitted under the GPL to combine a free program with a non-free program unless the entire combination is then released as free software under the GPL. This concept is often referred to as the so-called 'viral clause', i.e. free software 'infects' other software with its licensing terms, if a programmer chooses to use GPL'ed code [cf. Webe04, 48 et seq.].

The preamble of the GNU GPL clearly stipulates the aim of the license: 'The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU GPL is intended to guarantee freedom to share and change free software to make sure the software is free for all its users.' [cf. Free06b]. According to the FSF, the primary purpose of the GNU GPL is to preserve users' freedom to use, share and modify free software [cf. NoAu06c].

It is of the utmost importance to note that the GPL does not want to hinder that software licensed under the GPL is being commercially exploited. The GPL rather demands that anyone, who is exploiting a certain software, grants the same rights on the software as he himself received under the GPL. This means that any following user must have access to the source code, that he can either completely or partially modify the software and use it as part of a new product, for which those basic rules apply again. In order to duplicate, modify and distribute software it is necessary to have access to the source code. That's why it is necessary that any user of a GPL'ed software must be permitted to receive and read the GNU GPL to know about this right to source code [cf. Haar06, 24].

6.3. Development of the GPLv3

6.3.1. Steps of development of the GNU GPL

In 1989, the first version of the GNU General Public License was formulated by Richard Stallman and released by the Free Software Foundation. Only two years later, in 1991, the GPL was first modified. After Richard Stallman had been taking legal advice and collecting developers' opinions concerning the original version of the license, the FSF released version 2 of the GNU GPL. During the past 15 years, the GPL became the dominant free software license, with 70% of open source software licensed under it [cf. Hoch05, 10].

The GPLv3 is the first major update to the open source license since 1991 [cf. Hoch05, 10]. On January 16, 2006 the first public draft of the third version of the General Public License was released at a two-day Initial Conference [cf. NoAu06a]. This first draft was less controversial and more commercial-friendly than some expected [cf. Roon06, 12], but nevertheless led to a hot debate within the community. About half a year later, on July 27, 2006 the second discussion draft was released by the FSF. This second draft marks a halfway point of a yearlong public review for proposing changes and finalizing the GPLv3. The draft incorporates changes based on many of the suggestions for improving the license made by members of the free software community. Many of those suggestions have been discussed at international conferences held in the United States, Brazil and Spain [cf. NoAu06c], others within discussion committees [cf. Mant06b, R111].

By listening to people from around the world and incorporating suggestions made by the community, the FSF is working toward a license that acts consistently in many different legal systems and in a variety of situations [cf. NoAu06c]. Generally, the third version of the General Public License is much more complex and detailed than the previous version [cf. Mant06a, R42 et seq.]. The final version of the GPLv3 is expected by the spring of 2007 [cf. Hoch05, 10].

6.3.2. Objectives of the GPLv3

FSF founder Richard Stallman and Eden Moglen, general counsel for the FSF, are co-authors of the GPLv3 [cf. LaMo06], who both sticked to their principles of freedom. For

instance, they stipulate that software licensed under the GPL is not permitted to serve for unlawful interferences into privacy [cf. Haar06, 24].

15 years is an eternity in the world of software development, and the dramatically changed climate in which open source code is written and used calls for an update to the license [cf. Hoch05, 10]. The FSF claims that the changes incorporated in the drafts for the GPLv3 are being driven by a changing environment, which includes new restrictions and the worldwide expansion of the free software community [cf. NoAu06a].

One of the main reasons for the revision of the GPL is to more clearly block certain activities that hurt the community, but also to make the GPL more friendly and more responsive to the needs of businesses that develop and use free software [cf. Ries00]. Furthermore, it removes loopholes that could enable commercial vendors to hijack the GPL for their own purpose [cf. Roon06, 12]. Compatibility among GPL and other open source licenses can affect developers as well as users. If chunks of code are combined with others whose licenses do not allow for such mixing, sellers of these systems, and even users, could find trouble. That is why improving compatibility with other important free software licenses is another aim of the Free Software Foundation [cf. Hoch05, 10].

One problematic item of the Free Software movement is the issue of patents [cf. Ries00]. The GPLv3 contains a patent retaliation clause aimed at prohibiting developers from adding restrictions to their GPL-based products. [cf. Roon06, 12]. The first draft includes an automatic patent license free of charge. In this manner, the GPL permits patents but lifts its protection again by committing every patent holder to grant licenses free of charge to third parties. Thus, the GPL invalidates the fundamental idea of protection by patents, which is to grant the patent holder the exclusive right of disposal over the patent subject [cf. Haar06, 24].

6.3.3. The GNU GPL and Digital Rights Management (DRM)

Another hot topic concerning the revision of the GPL is Digital Rights Management (DRM) – or 'Digital Restrictions Management', as the FSF puts it [cf. NoAu06a].

6.3.3.1. Characteristics of Digital Rights Management

Digital Rights Management systems are used to protect high-value digital assets and control their distribution and usage. DRM systems are intended to offer a persistent content protection against unauthorized access to the digital content, limiting access to only those with the proper authorization. Such a system should be flexible to manage user rights for different kinds of digital content (e.g. images, music files, digital books) across different platforms (e.g. laptops, PDAs, mobile phones) and control access to content delivered on physical media or any other distribution method (e.g. DVDs, CD-ROMs) [cf. Liu03].

The core concept of Digital Rights Management is the use of digital licenses, which specify certain usage rules for a digital content. Those usage rules can be defined by a range of criteria, such as frequency of access, expiration date, restriction of transfer to other devices, copy permission, etc., and can be combined to enforce certain business models. When applying a DRM system, the consumer purchases a license granting certain rights to him instead of buying the digital content. Through digital licensing, content providers can gain much more control over what the consumer can do with the content [cf. Liu03].

Even though different vendors have different DRM implementations, names and ways to specify the content usage rules, the basic DRM process is the same in either case, involving four parties: the content provider, the distributor, the clearinghouse and the consumer. First of all, the content provider holds the digital rights of the content and wants to protect these rights. The distributor provides distribution channels and receives the digital content from the content provider. The consumer uses the distributor's system to consume the digital content by retrieving the content through the respective distribution channel and then paying for the license. Finally, the clearinghouse handles the financial transaction for issuing the digital license to the consumer and pays royalty fees to the content provider and distribution fees to the distributor accordingly. In addition, the clearinghouse is responsible for logging license consumptions for every consumer. [cf. Liu03].

Figure 8 shows a typical DRM model and how the common parties and components interact in such a model.

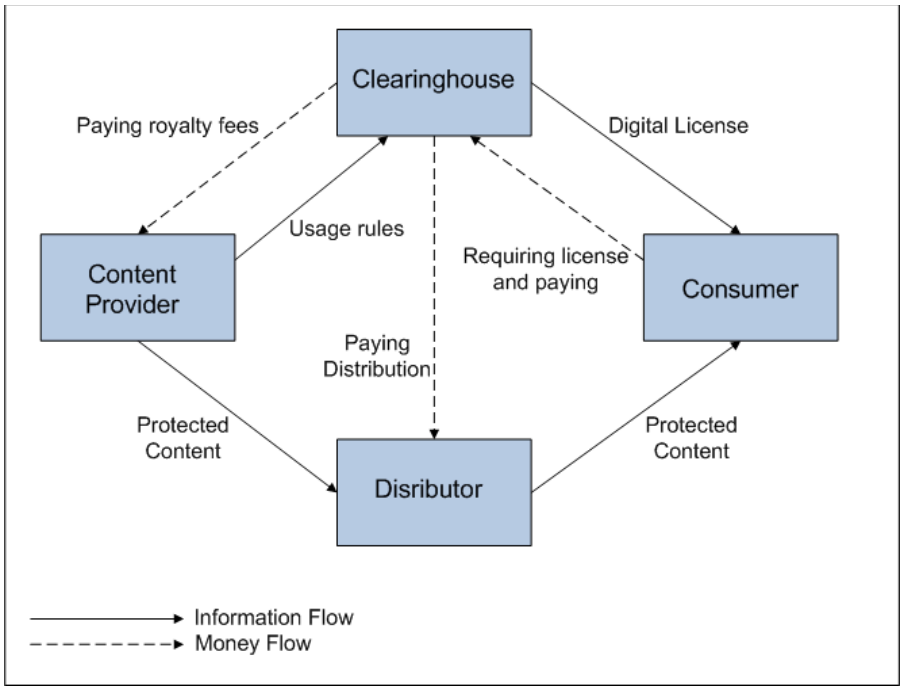

Figure 8: DRM model [cf. Liu03]

6.3.3.2. Digital Rights Management and the Free Software Foundation

Eben Moglen, general counsel for the FSF states that the new version of the GPL, the most widely used open source license, takes a highly aggressive stance against the digital rights management software that's widely favored in the entertainment industry [cf. LaMo06]. According to the FSF, DRM is fundamentally incompatible with the purpose of the GPL, which is to protect users' freedom [cf. NoAu06a] as DRM technology places limits on how consumers can play movies, music or other digital content. DRM systems that take control out of people's hands or violate their privacy do not respect the rights of free software users and therefore are in conflict with the forthcoming GPL provisions [cf. LaMo06].

As some countries have adopted laws prohibiting software that enables to escape from DRM, the GPLv3 ensures that the software it covers will neither be subject to, nor subject other works to, digital restrictions from which escape is forbidden [cf. NoAu06a]. The current draft states that GPL software cannot use digital restrictions on copyright material unless users can control them [cf. LaMo06]. This means that the license does not prohibit the implementation of DRM, but prevents DRM features that can not be removed [cf. NoAu06c]. The FSF says that those clauses restricting DRM only clarify points that were already implied in previous drafts [cf. NoAu06d].

As outlined above, the new version of the GNU General Public License includes anti-DRM provisions that put it in conflict with movie studios. The planned anti-DRM changes to the GPL are significant because the entertainment industry regularly uses Linux-powered computers in the production process, notably for special effects and animation [cf. LaMo06].

6.3.3.3. Prospects for the GPLv3

Peter Brown, executive director of the FSF, emphasizes that any open source product licensed under GPLv2 will have to be relicensed for Version 3. It's the software developers' decision only to do so [cf. Hoch05, 10]. The bottom line is that the GPLv3 will be only as important or powerful as the software that developers and vendors release under it. If a developer or vendor does not like the GPLv3, they don't have to use it for their projects and have the option of not distributing or building upon the works of developers who have chosen to use the license [cf. Broo06, 18].

As the drafts for the new version of the GPL have caused a lot of controversy, there seem to be only two choices for the FSF concerning the future development of the license. One is to opt for a GPL that maximizes freedom over the business and development model values of other current GPL stakeholders – and maybe end up with a license that nobody uses. The second one is to scale controversial provisions back and hang onto the major free software projects that put the GPL on the map in the first place [cf. Broo06, 18].

7. Stakeholders of free software

As there is a multitude of individuals, companies, institutions and other groups who take a stock in either the usage, the further development or also the crowding out of free software, a complete analysis of all stakeholders of free software would certainly go beyond the scope of this paper. The author will therefore focus on two specific groups of stakeholders: programmers on the one hand and users on the other hand. First of all, the factors of motivation for programmers and developers of free software are analyzed in order to point out the most important reasons why programmers engage in the development of free or open source software. Subsequently, the advantages as well as the disadvantages for users of free software products are pointed out to give a review on how users can benefit or disbenefit from using free software instead of proprietary software.

7.1. Factors of motivation for programmers

When it comes to the development of free software or open source software, it is obvious that the existence of a certain community plays a crucial rule for developing such types of software where source code is freely available and released for modification by anyone. Personal efficacy not only benefits from, but positively requires, a set of cooperative relationships with others. Thus, a community empowers the individual to help himself [cf. Webe04,145]. A high level of practical knowledge, support and qualified collaborators are only some of the advantages of an active community. A community concerned with developing free or open source software is often much more diverse than a community occupied with developing non-free software. Factors that positively contribute to the formation of a community are freedom, an open software development process and a broad economic relation with free as well as commercial support [cf. Reit04, 88].

Individual programmers often use their expertise from their main occupation, i.e. a lot of programmers engaged in developing free software have a full-time job dealing with the development of either free but also non-free software [cf. Reit04,88]. The ability of open software development to cumulate and utilize the collective knowledge and expertise of thousands of developers is an impressive feature. The high level of connectivity between

the programmers plays an important rule for the speed of both the development and the distribution of free and open source software [cf. Möll05, 70].

[Gras04, 252] emphasizes that there are several individual and social factors of motivation:

- intellectual challenge
- creativity and pride for achieving something
- implement software that fulfills somebody's own requirements of style and quality
- social contact with people who share the same ideas and interests
- fame
- advancement of the collective identity

[Möll05, 63 et seq.] adds the following factors that motivate programmers to develop free and open source software:

- political idealism
- joy due to working on an own project without instructions from a superior
- ego-satisfaction, i.e. a good feeling to see how other people use the own program
- reputation within the community
- curiosity
- charity, i.e. acting beneficially for the public
- thankfulness: After having used software developed by others for many years the development of an own program offers the possibility to give something back to the community.

A shared belief is that experimentation is the highest form of human behavior. To try new things that challenge one's skills and the skills of others is not just a tool for individual learning and development but also a contribution to the community. In addition, the high-intensity race for ego-boosting explains some of the energy that developers devote to open source work [cf. Webe04, 145 et seqq.].

Individual motivations do not make up anything like a full explanation for the success of open source. The organization of the community that provides a necessary macrostructure to the open source process is not simple, even if the individual decisions to write open

source code may be 'simple' from an economic standpoint [cf. Webe04, 149].

Companies frequently engage in the development of free and open source software due to own requirement. If a company needs a specific computer program for improving internal operational procedures which is not available on the market yet, it often chooses in-house development [cf. Reit04, 88].

7.2. Advantages and disadvantages for users

7.2.1. Advantages of free software

Free software has various advantages for users. First of all, while users have to pay high license fees for proprietary software, there is no license fee at all for free software which also leads to wide distribution of these programs [cf. Facu06]. In contrast to proprietary software, free and open source software is available for limited cost or totally free of charge. It can be freely and legally copied and used on an unlimited number of computers.

In general, free software is more stable than proprietary equivalents, because the possibility of having access to the source code makes it much easier to correct errors. Due to the wide user community, errors or security leaks can be found faster and can be fixed by skilled users who are able to modify the source code. This leads to a faster improvement in quality compared to proprietary software where users reporting bugs to the company can only wait for a new update of the program. Another advantage of the availability of source code is the adaptability of a program to the user's individual needs including functional extensions and adjustments to new technologies. Free software offers users a high degree of flexibility because programs can be adapted in order to develop specific solutions. The large and active communities of free and open source software projects guarantee the availability of support [cf. Facu06].

Another advantage of free software is reusability. Due to its free availability, free software is reusable both in terms of reusing the license and in terms of reusing the source code. While the first results in cost savings, the second results in higher reliability through testing, more feedback, etc. [cf. Pent05, 22].

Interoperability, which is due to the use of open standards, is another characteristic of open source software. Furthermore, the adaptability of open source software combined with the fact that it is developed by using open standards facilitates its integration with other software, either with other open source software or with proprietary software that uses the same open standards. Free and open source software systems usually have modular designs with well-defined interfaces. Modularity is a basic prerequisite for the distribution of tasks within the community of developers and facilitates software maintenance. In addition, open source software is usually developed for a variety of software platforms and hardware platforms and is generally more portable than proprietary software [cf. Pent05, 20 et seqq.].

Access to the source code of free software programs is also a guarantee of permanence and independence of users from software publishers. Users of proprietary software have very little influence on further developments, whereas users of free software can modify the source code at any time, which makes them independent from software manufacturers. Under a suitable license with the copyleft paradigm, a user is obliged to pass any error fixes and developments back to the community under the same license agreement and without any license fee, which guarantees the return of developments made by any third party.

7.2.2. Disadvantages of free software

One disadvantage related to the use of free or open source software is the risk of project termination. Since open source software is generally developed on a voluntary basis, an open source project can be terminated prematurely if there are no developers left who are interested in maintaining the software. Nevertheless, in the case of major open source projects where the number of participating developers is very large this situation is unlikely to arise. In addition, even if community support for an free software product ceases, users using the software always have the option of maintaining it themselves if they are skilled enough or finding somebody to maintain it for them [cf. Pent05, 22].

Concerning open source software, a lack of documentation can frequently be identified, which is mainly due to the fact that programmers involved in open software development are developing and debugging software rather than writing documentation. Users who are

new to the open source field and especially those users who are generally unexperienced with information technology, especially software, may find it difficult to obtain useful information as the available websites where open source projects are hosted are usually designed for developers rather than for users. The lack of documentation is often accompanied by poor usability. Compared to proprietary software, the use of free or open source software often requires more user effort and expertise. Particularly non-experts may also suffer from non-intuitive or non-existing graphical user interfaces (GUIs) and the lack of user-friendliness, but the usability of open source software is increasing constantly. The fast and continuous development of this type of software requires a constant effort to keep up to date, which may also be difficult for unexperienced users [cf. Pent05, 22 et seq.].

One of the major disadvantages of free software commonly mentioned is the absence of a claim for support on the part of the producer, i.e. the author of the software. But the fact that such a claim is indeed missing is misleading because even suppliers of proprietary software generally offer their help only against payment. The already mentioned lack of documentation of free software point to the heart of the problem: if the only help available is the source code, it can easily lead to helplessness for unexperienced users. Therefore more and more providers of open source software provide a set of FAQs (frequently asked questions) on their project websites dealing with the users' most common problems and discussion forums where users can ask for help or search for existing troubleshooting.

Free and open source software is delivered without any warranty or liability, which means that nobody can be held legally responsible if the software causes any damage. As disclaiming warranty and liability for a product is legally forbidden in some countries, open source software is often regarded as a donation or gift in these countries in order to make open source software compatible with the existing laws [cf. Pent05, 23].

8. Conclusion

The increasing availability and popularity of free software and open source software has significantly changed the rules of the game. As the ideas of free software have risen in publicity, more and more users have become interested in using this kind of software for personal use. But also companies and public authorities have started to utilize free software more often and benefit from the advantages stemming from the four freedoms that form the basis of free software. Those fundamental freedoms, which were first formulated by Richard Stallman, comprise the freedom to run, copy, distribute, study, modify and improve software.

For an ordinary user who is mainly interested in receiving an appropriate computer program to solve a certain problem, the terms 'free software' and 'open source software' mostly signify the same. For such a user, these terms imply that he does not have to pay any license fee, that he can run the program on several different devices, that he can copy and distribute the program or – in case he has the necessary skills to do so – that he can modify the software and adapt it to his individual needs. By contrast, those two terms do certainly not mean the same for advocates of either the Free Software Foundation or the Open Source Initiative. For advocates of Richard Stallman and the FSF, the idea of free software involves much more than only the principle of open source code. It goes much further by saying that software should not have owners because most of them aim to withhold a software's potential benefit from the rest of the public by placing restrictions on how this software can be used, modified or redistributed. But as the members of the Free Software movement and the Open Source movement disagree on the basic principles, but agree more or less on the practical recommendations, they can and do work together on many projects after all.

Richard Stallman, the figurehead of the Free Software movement and probably the biggest asserter of the ideas of free software, has established the Free Software Foundation to further promote his visions and to support the GNU project which was equally fathered by Richard Stallman as well. Even though the advocates of the GNU project did not succeed in achieving the main aim of the project, which was to develop a complete Unix-like operating system that was free software, they probably made the development of the Linux

operating system possible by developing and providing the various components of the GNU system.

The concept of copyleft, which was mainly characterized by Richard Stallman, provides the foundations for the GNU General Public License, which has become the most widespread and important free software license. But after 15 years of leaving the license unmodified, the dramatically changed environment in which free and open source software is written and used has finally called for an update to the license. The development of the new version of the GNU GPL, the GPLv3, is characterized by a yearlong public review for proposing changes and finalizing the license. The drafts for the GPLv3 have already incorporated changes based on numberless suggestions made by members of the free software community.

As Richard Stallman once said, 'competition is not harmful; the harmful thing is *combat*' [cf. Stal02, 130]. An active community keen on experimenting is vital for successfully developing free or open source software. As a large number of committed programmers are working together to develop a certain computer program, they all contribute to the project with their individual knowledge, experience, skills and ideas. In the near future, the key to success is possibly to more an more proceed to open software development benefiting from the specific dynamics and cooperation of such a community rather than sticking to closed development of software.

9. Bibliography

[Broo06] Brooks, Jason: Free to be GPL 3? In: eWEEK, 2006-08-07, pg. 18.

[Gall06] Galli, Peter: GPL 3 draft revives license debate. In: eWEEK, 2006-07-17, pg. D6-D8.

[Gras04] Grassmuck, Volker: Freie Software. Zwischen Privat- und Gemeineigentum. Bundeszentrale für politische Bildung (bpb), Bonn 2004.

[Haar06] Haar, Tobias: Diskussionsfutter – Entwurf der GNU GPL Version 3. In: iX – Magazin für Informationstechnik, 03/2006, S. 24.

[Hoch05] Hochmuth, Phil: Open source GPL to get major revision. In: Network World, Vol. 22, Iss. 48, 2005-12-05, pg. 10.

[Kard04] Karduck, Achim P.: Free and Open Source Software: Einfluss auf ICT-Entwicklungsstrategien. In: HMD: Praxis der Wirtschaftsinformatik, Band 41 (2004) 238, S. 5-18.

[Mant06a] Mantz, Reto: GPL: Version 3 zur Diskussion. In: Computer und Recht (CR), 4/2006, S. R42-R43.

[Mant06b] Mantz, Reto: Neue Entwürfe von GPLv3 und CCPLv3 vorgestellt. In: Computer und Recht (CR), 10/2006, S. R111.

[Möll05] Möller, Erik: Die heimliche Medienrevolution – Wie Weblogs, Wikis und freie Software die Welt verändern. Heise, Hannover 2005.

[NoAu06a] No Author: Free Software Discussion Underway with GPLv3. In: TechWeb, 2006-01-18.

[NoAu06b] No Author: Linux Creator Calls GPLv3 'Crusade'. In: TechWeb, 2006-02-08.

[NoAu06c] No Author: Linux Leader Takes Aim At Free Software Movement. In: TechWeb, 2006-07-31.

[NoAu06d] No Author: Free Software Advocates Defend GPLv3. In: TechWeb, 2006-08-03.

[Pent05] Pentas, Parachos: Evalutation of open source content mangagement systems based on the specific requirement of the Intercultural Academic Network. Master thesis, University of Applied Sciences Darmstadt, 2005/06.

[Reit04] Reiter, Bernhard E.: Wandel der IT: Mehr als 20 Jahre freie Software. In: HMD: Praxis der Wirtschaftsinformatik, Band 41 (2004) 238, S. 83-91.

[Roon06] Rooney, Paula: GPLv3 Proposal Extends Compatibility. In: CRN, 2006-01-23, pg. 12.

[Stal02] Stallman, Richard M.: Free Software, Free Society: Selected Essays of Richard M. Stallman. GNU Press, Boston, MA 2002.

[StLa04] St. Laurent, Andrew M.: Understanding Open Source and Free Software Licensing. O'Reilly, Sebastopol, CA 2004.

[Webe04] Weber, Steven: The Success of Open Source. Harvard University Press, Cambridge, Mass et al. 2004.

Online Sources:

[Bund06] Bundeskanzleramt Österreich: Rechtsinformationssystem Bundesrecht - Urheberrechtsgesetz § 60, 2006-01-01,
http://ris.bka.gv.at/bundesrecht/,
visit on 2007-01-01.

[Drei96] Dreier, Thomas: Copyright issues in a digital publishing world.
Joint ICSU Press/ UNESCO Conference on Electronic Publishing in Science, UNESCO, Paris, 19-23 February 1996,
http://citeseer.ist.psu.edu/347622.html,
visit on 2007-01-02.

[Facu06] Faculty of Electrical Engineering / Department of Communication Systems: The Open Source Approach.
http://cs.fernuni-hagen.de/activities/projects/uk_index_open_source.html,
visit on 2007-01-03.

[Free05a] Free Software Foundation: The Free Software Definition.
http://www.fsf.org/licensing/essays/free-sw.html, 2005-02-12,
visit on 2006-12-30.

[Free05b] Free Software Foundation: What is Copyleft?
http://www.fsf.org/licensing/essays/copyleft.html#WhatIsCopyleft, 2005-02-12
visit on 2006-11-24.

[Free05c] Free Software Foundation: Diagram of the different categories of software.
http://www.fsf.org/licensing/essays/category.jpg/view, 2005-02-12,
visit on 2006-12-22.

[Free05d] Free Software Foundation: Categories of Free and Non-Free Software.
http://www.fsf.org/licensing/essays/categories.html, 2005-02-12,
visit on 2006-12-20.

[Free05e] Free Software Foundation: GNU General Public License.
http://www.fsf.org/licensing/licenses/gpl.html/view, 2005-05-02,
visit on 2006-12-29.

[Free06a] Free Software Foundation: The GNU Manifesto.
http://www.gnu.org/gnu/manifesto, 2006-05-22,
visit on 2006-12-08.

[Free06b] Free Software Foundation: GPLv3, 2nd discussion draft.
http://gplv3.fsf.org/gpl-draft-2006-07-27.html, 2006-07-27,
visit on 2006-12-30.

[Free06c] Free Software Foundation: Various Licenses and Comments about Them.
http://www.fsf.org/licensing/licenses/index_html, 2006-09-15,
visit on 2007-01-02.

[Free06d] Free Software Foundation: The GNU Operating System.
http://www.gnu.org/, 2006-12-20,
visit on 2006-12-29.

[Free06e] Free Software Foundation: Overview of the GNU System.
http://www.gnu.org/gnu/gnu-history.html, 2006-08-20,
visit on 2006-12-29.

[Free06f] Free Software Foundation Europe: What is the GNU project?
http://www.germany.fsfeurope.org/documents/gnuproject.en.html, 2006-12-19,
visit on 2007-01-02.

[GnuE06] GNU España: Free Software Concept Map.
http://es.gnu.org/~reneme/map/map-en.png, 2006-08-05,
visit on 2006-12-29.

[Info07] Information Service Board: Policy Definitions.
http://isb.wa.gov/policies/definitions.aspx,
visit on 2007-01-02.

[LaMo06] LaMonica, Martin: GPL 3 to take hard line on DRM.
http://news.zdnet.co.uk/itmanagement/0,1000000308,39247976,00.htm, 2006-01-19,
visit on 2006-11-07.

[Linu06] Linux.bz: Freedom: as in Speech (libre) and as in Beer (gratis).
http://linux.bz/index.php/markus/freedom/en,
visit on 2007-01-02.

[Liu03] Liu, Qiong et al.: Digital Rights Management for Content Distribution.
http://www.itacs.uow.edu.au/research/smicl/publications/aisw2003.pdf, 2003,
visit on 2007-01-01.

[NoAu06e] No author: Richard Stallman.
http://home.earthlink.net/~johnrpenner/Images/Entail-Stallman-SamOgden.jpeg,
visit on 2007-01-04.

[Ries00] Ries, Eric: Sneak preview of GPL v.3. Part 2: System libraries and patents. http://www.newsforge.com/article.pl?sid=00/12/14/1910252, 2000-12-14, visit on 2006-11-11.

[Stal92] Stallman, Richard M.: Why Software Should be free. http://www.gnu.org/philosophy/shouldbefree.html, 1992-04-24, visit on 2006-12-01.

[Stal98] Stallman, Richard M.: The GNU Operating System and the Free Software Movement – The First Software-Sharing Community. http://www.eroj.org/linux/movement.htm, 1998, visit on 2006-11-30.

Further reading:

Free Software Foundation:	http://www.fsf.org/
Free Software Foundation Europe:	http://www.germany.fsfeurope.org/
Free Software Foundation India:	http://fsf.org.in/
Free Software Foundation Latin America:	http://www.fsfla.org/
GPLv3:	http://gplv3.fsf.org/
Open Source Initiative:	http://www.opensource.org/
Richard Stallman (Personal Homepage):	http://www.stallman.org/
The GNU Operating System:	http://www.gnu.org/
U.S. Copyright Office:	http://www.copyright.gov/