Open Source

Educational Technology Expertise Centre OTEC
Open University of the Netherlands

Open source

History and relevance for the Technology Development programme

COLOPHON

Title:	Open source History and relevance for the Technology Development programme
Author(s):	Harrie Martens Marc Verhooren
Project:	WP2
Project manager:	Ing. H.F. Vogten
Project members:	E.J. van den Berg dr.ir. F.M.R. Brouns drs. H.J.H. Hermans H.G.H. Martens Ir. G.W. van der Vegt Ing. M. Verhooren
Programme:	Technology Development Programme
Programme chair:	Prof.dr. E.J.R. Koper
Programme assistant:	Mieke Haemers
Publication date:	30 March 2004
Document reference:	U2003-6001 MMO
Distribution:	OTEC

Educational Technology Expertise Centre (OTEC)
Open University of the Netherlands

Open Source

History and relevance for the Technology Development Programme

OTEC report series

The Open University of the Netherlands develops higher distance education and is a central partner in a consortium for the renewal of higher education. Educational technological innovation is one of the main fields of interest. The educational and educational technological expertise of the Open University of the Netherlands are bundled in the Educational technology expertise centre (OTEC). This centre is involved with tasks concerning the development, innovation, research and evaluation of the Open University and its consortium partners. These tasks are performed in close collaboration with directorates and faculties of the Open University and/or its consortium partners.

OTEC publishes a report series. This report is part of that series.

Development Programme reports and some documents can be obtained from:

Open University of the Netherlands
Secretary Development Programme
P.O. Box 2960
6401 DL Heerlen
the Netherlands
Tel. +31 45 5762624
Fax. +31 45 5762800
Internet: http://www.ou.nl/otec

Table of contents

1. Introduction

The Open University of the Netherlands uses since its start in the early 1980's computers to support its own organization as well as to enhance the learning experience of its students. Over the years all different kinds of hardware and software platforms have been used. But there has always been one common denominator; all software used was closed, proprietary software. This included all the software which was developed in house or by subcontractors.
The technology development programme of OTEC recently expressed an interest in looking into other kinds of software like non-closed, non-proprietary software. This kind of software is generally referred to as 'Free Software' or as 'open source software'.
But new ideas inevitably bring new misconceptions with them. Many people have different perceptions on the ideas behind free software or open source software. This report will try to clarify these issues by clearly defining and explaining these terms. Furthermore it will show the advantages and disadvantages of the open source definition. It will also discuss the open source development method, called 'Bazaar style' and give criteria on when to choose or not to choose for open source.
Finally, this report will look at the way the technology development programme benefits from open source software and open source development.

2. History

To understand the history of free software and open source it is imperative to know the history of GNU/Linux because the two topics are closely intertwined.

2.1 The history of Unix

In 1965 Bell Telephone Labs, the Massachusetts Institute of Technology and General Electric started working Multics. This was one of the first multi user computer systems and it is still in use today.
In 1969 Bell Telephone withdrew from the project leaving two software engineers, Ken Thompson and Dennis Ritchie, unable to play their favourite pastimes. This game, called Space Travel, works only on the Multics platform (Ritchie, 1984). And by leaving the project the two had no access to a Multics system. They decided to port the game to another computer, a PDP-7 computer. During this port they eventually developed a rudimentary operating system they named Unics, as a pun on Multics. Somehow, the spelling of the name became Unix.

Because their initial project was to port a game from one computer platform to another they decided to make Unix portable, to make it easier to adapt the operating system to different computers. In order to create this portable operating system they first created a portable programming language, called C. C was an intermediate language between the low level assembly languages, which give the programmer access to low level hardware facilities but at the cost of portability and between high level languages like FORTRAN and COBOL. These latter were portable but didn’t give the programmer access to the low level facilities, needed to implement an operating system.

Over time the interest in their work grew and Thompson and Ritchie made copies of Unix freely available to other programmers. In turn these programmers revised and improved Unix, and even more importantly sent their changes back to its creators. By incorporating the best changes Unix matured as an operating system. Over time different versions of Unix began to emerge. And in 1984, AT&T, the parent company of Bell Labs, began selling its own version of Unix, known as System V. AT&T claimed Unix as its intellectual property and started asking a considerable license fee. The other parties that had implemented their own versions of Unix copied this approach and also started asking a fee for their licenses.

2.2 Free Software Foundation

Understandably, those who had contributed improvements to Unix considered it unfair for AT&T and others to appropriate the fruits of their labours. This concern for profit was unlike the democratic, share-and-share-alike spirit of the early days of Unix. Some, including MIT scientist Richard Stallman, yearned for the return of those happier times and the mutual cooperation of programmers that then existed. So, in 1983, Stallman launched the GNU (GNU's not Unix) project, which aimed at creating a free Unix-like operating system. Like early Unix, the GNU operating system was to be distributed in source form so that programmers could read, modify, and redistribute it without restriction. Stallman's work at MIT had taught him that, by using the Internet

as a means of communication, programmers the world over could improve and adapt software at incredible speed, far outpacing the fastest rate possible using traditional software development models, in which few programmers actually see one another's source code.

As a means of organizing work on the GNU project, Stallman and others created the Free Software Foundation (FSF), a non-profit corporation that seeks to promote free software and eliminate restrictions on the copying, redistribution, understanding, and modification of software. Among other activities, the FSF accepts tax-deductible charitable contributions and distributes copies of software and documentation for a small fee, using this revenue to fund its operations and support the GNU project.

2.2.1 Linux

Work on the GNU system and many of its libraries and tools continued for many years and by 1990 the system was almost complete. The only missing major component was the kernel. The programmers had decided to create a kernel called the GNU HURD. The creation and debugging of the HURD took a lot longer than expected and in 1991 another kernel became available. It was created by Linus Torvalds and it was a Unix-compatible kernel called Linux. The GNU system was combined with the Linux kernel and by 1992 there was a complete free operating system called GNU/Linux. When nowadays people talk about Linux, most of the time it is actually GNU/Linux they are talking about, because, as indicated, Linux itself is only a kernel, not a complete operating system.

2.3 Open Source Initiative (OSI)

In the beginning of 1998 Netscape announced that it planned to give away the source of its browser. Following that announcement, a group of people sat together in a strategy session to discuss the opportunity to bring the practical benefits of sharing software source code to the attention of the software industry. They came up with the label 'open source' and formed the Open Source Initiative (OSI).

They decided to leave the whole ethical question of free software out of attention and focus on the pragmatic benefits of sharing source code. For this reason the Free Software Foundation decided to distance itself from the Open Source Initiative.
The Open Source Initiative is a non-profit corporation that accepts tax-deductible contributions. Its most famous member is probably Eric Steven Raymond, who wrote several books and publications about open source. His book 'The Cathedral and The Bazaar' describes the difference between the classic software development model (the Cathedral model) and the open source development model (the Bazaar model) (Raymond, 2000).

Where the Open Source Initiative focuses on the practical implications of software development, the Free Software Foundation mainly focuses on the ideological aspects (freedom) of software development.

3. Free Software

3.1 Definition

The Free Software Definition is used by the Free Software Foundation to express clearly what is meant by free software.

The word ‘free’ in ‘free software’ can and will lead to many misunderstandings about what is meant by free software. Free software is not about price, it is about liberty. It’s ‘free’ as in ‘free speech’, not as in ‘free beer’.

Software is ‘free software’ if it grants the user:

1. the freedom to run the program for any purpose
2. the freedom to study how the program works and to modify it
3. the freedom to redistribute copies
4. the freedom to modify the program and release it to the public.

Please note that in order to comply with freedoms 2 and 4, access to the program’s source code is necessary.

The freedoms above mean that you do not have to ask or pay for permission to redistribute, modify, run or sell copies of free software. Asking for a fee for distribution of the software is allowed. However, the person that receives the software is then allowed to do with it whatever he wants, e.g. distribute the software for free on a website. It is also allowed to modify the software without reporting any modifications back. You are also not required to publish the changes. The freedom to redistribute copies means that you must include binaries or executables as well as the source code of the program, given of course that it is possible to create binaries or executables.

3.2 Licenses

Free software is protected legally by licenses. The Free Software Foundation decides whether a certain licence is a free software license based on the criteria mentioned above.

Almost all the software of the GNU project is protected by licenses that use ‘copyleft’. Copyleft means that it is prohibited for redistributors of the software to add restrictions to the distribution terms of changed or redistributed software. This means that free software will always be free software, even if it is modified or redistributed. The following statement comes from the Free Software Foundation’s web site and explains copyleft:

> ‘To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code *or any program derived from it* but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable.’

The most prominent 'copyleft' licenses are the GNU General Public License (GNU GPL) and the GNU Lesser General Public License (GNU LGPL). The GNU GPL protects free software in the real spirit of the Free Software Foundation's ideas. The GNU LGPL is less restrictive, in the way that it permits linking with non-free modules. The GNU GPL is included in annex I.

One thing all these licenses have in common is the disclaimer. They deny all warranties with the intent to protect the software creator from any liability connected with the program. Since the program is often being given away at no cost, this is a reasonable requirement – the author does not have a sufficient revenue stream from the program to fund liability insurance and legal fees.

So this is done to make sure that free software contributors do not get frustrated by legal issues and stop contributing to the free software community.

3.2.1 Types

The Free Software Foundation distinguishes 3 types of software licences:

1. Free software, GPL-compatible:
   All these licences reflect the same principles the GPL does. This means that a software module licensed with one of these licenses can be combined with software released under another GPL-compatible license without violating one of these principles. Licenses include:
   - GNU GPL
   - GNU LGPL
   - the modified BSD license (an advertising clause was deleted)
   - W3C Software Notice and License
   - Zope Public License version 2.0
   - Netscape Javascript.

2. Free software, GPL-incompatible:
   One cannot combine modules under these licenses with modules under GPL licenses without restricting the GPL, which is not allowed. In most cases these are GPL-incompatible, because the Free Software Foundation does not agree with some parts of the license. Licenses include:
   - the original BSD license (there is an advertising clause in it, which causes practical problems, according to the Free Software Foundation)
   - Apache license, versions 1.0 and 1.1
   - Zope Public License version 1 (practical problem like the BSD)
   - PHP License version 3.0
   - Mozilla Public License.

3. Non-free software:
   License does not qualify as free software. Licenses include:
   - Apple Public Source License
   - Open Public License
   - YaST license.

Besides software licenses the Free Software Foundation recognizes also documentation licenses.

3.2.2 Public Domain

‘Public domain’ is a legal term and means, precisely, ‘not copyrighted’. So a public domain program is a program for which the author has deliberately surrendered his rights. It does not come with a license. One is free to do with it what one likes. You can treat it as your personal property. You could remove the author’s name or release the program under a (new) license.

Free software is often confused with public domain software. The difference is that free software has a license that protects the author’s rights while public domain software has not.

4. Open source software

4.1 Definition

Just like the Free Software Definition there is an Open Source Definition (OSD). The Open Source Definition is composed by Bruce Perens and is derived from the Debian Free Software Guidelines (Perens, 1998).

The Open Source Definition wants to make sure that the programmers are assured of the following rights:

- The right to make copies of the program and to distribute these copies
- The right to access the program's source code
- The right to make improvements to the program.

The Open Source Definition states nine criteria. These are distribution terms that software must comply with in order to be called 'open source software'. These criteria state that one must be free to make copies of the software and sell or give these away. Access to source code must also be free and must be distributed with the initial work. Modifications must be allowed and the programmer must be allowed to distribute his derived work under the same license terms. However he is **not required** to distribute under the same license terms. An open source license may restrict source-code being distributed in modified form only if the license allows the distribution of patch files that modify the source code at build time. This rule is thought up to protect the integrity of source code. Authors were afraid one would mistake the modified source for their original work which could reflect poorly on their work. By distributing these patch files a clear distinction is made between the original and the modified code. The license must also not restrict persons, groups or certain fields of endeavour from making use of the program. Furthermore the license must apply to everybody using it and must not be specific for the use with a certain distribution. The license cannot place restrictions on other software that is distributed with the licensed software.

4.2 Licenses

In all cases, licenses that qualify as free software licenses, also qualify as 'OSI Certified open source software'. So these licenses include the well-known GNU GPL, GNU LGPL, BSD, etc. It is not true that all OSI approved licenses automatically qualify as open source licenses, because the OSI is a little less restrictive in its criteria. The Free Software Foundations has some issues with certain licenses that the OSI has not. So, for that reason, the Apple Public Source License as well as some other licenses qualify as open source license but not as a free software license.

4.3 Comparison with free software

The main difference between free software and open source software is the difference in attitude. The free software movement places the primary emphasis on the moral

and ethical aspects. They say that free software is about freedom, the freedom to run, copy, distribute, study, change and improve the software. The practical advantages and technical progress are desirable by-products of their standard. Open source's main focus however is on the practical benefits of free software. They believe that by making source code available for everyone you'll get a better, more stable, cheaper (as in cost of ownership) software product. They were afraid that the (conservative) business world would be scared off by the strong advocacy for freedom by the Free Software Foundation. In their article about the history of the OSI is written:

'*We realized it was time to dump the confrontational attitude that has been associated with free software in the past and sell the idea on the same pragmatic, business-case grounds that motivated Netscape*'.

When OSI was founded in 1998 Richard Stallman and his Free Software Foundation were asked to adopt the term, but declined because of the difference in attitude. In his article 'why free software is better that open source' he writes:

'*The fundamental difference between the two movements is in their values, their ways of looking at the world. For the open source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, open source is a development methodology; free software is a social movement. For the open source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution*.'

Although the groups disagree on the basic principles, they agree more or less on the practical recommendations that come with free software/open source software.

OSI is a little less restrictive regarding the acceptance of software licenses. Both groups however recommend the General Public License as the best license should you want to make your software free software or open source software.

The remainder of the report will focus on open source because this is the more widely adopted approach to free software/open source. Because of the more pragmatic nature of the open source approach it is favoured by the IT community which considers the free software approach to be too ethical and too confrontational.

5. Advantages/Disadvantages of open source

The following paragraphs list the advantages and disadvantages of the use and creation of open source software as they can be found on many websites and in many magazine articles. This is a compilation of the most heard and read issues. The issues will be briefly discussed.

5.1 Advantages

Lower price
Open source software is usually for free. For the software itself no (licensing) fee has to be paid. In many cases a distributor will do some extra work, like creating manuals, collecting various pieces of software together or make an easy and user-friendly installer for the software. For these services usually a fee will be charged, which is mostly a very reasonable one. A consumer can choose in this case either to take the software as-is for free, or pay a small fee and get many extras. Good examples of this approach are the various Linux distributions, like Red Hat and SuSE.
Making software available for free is not exclusively a feature of open source software. Sometimes a firm releases its software for free (e.g. to reach a bigger audience) without making the source available.

Higher reliability
It is a much heard opinion that open source software is more reliable than closed-source software in the sense that it usually contains fewer bugs. The reason for this is that with open source there is much more room for peer review. When there are more people looking at the code, there are more people who will find bugs. It is also argued that in an open source setting, there is always someone who will find the problem of his interest and there are more people that can come up with a solution for the problem. This argument only holds true if a sufficiently large developer community has been established around the project.

Customisability
Open source software gives the user the possibility to fix bugs himself or to customise software to make it more suitable to his own needs. He could make these changes himself or hire an IT firm that can make the changes for him.

Future-proof
The fact that the source code is open for the software means that it is always possible for someone to learn to know the code and the program in order to provide support. This is a very important feature should the vendor of the software decide not to support the software anymore or should the vendor go bankrupt.

Avoiding proprietary lock
Anyone with enough expertise can learn how the software works and can make changes to it or provide support for it. This avoids that a consumer is forced to hire the vendor of the software for changes and support.

5.2 Disadvantages

No licensing fees
For open source software no licensing fee has to be paid. It is essentially for free. This is a problem for a company that wants to make money from its software. As discussed, a way to make money from open source software is to provide additional services like distributions, manuals, support etc. This does not mean that a company loses money by making open source software, because often the company gets public funding for this type of software.

Quality and Support
It is sometimes said that due to the legal obligations that a commercial firm has, the quality and support for the software are higher. The fact that such a company can be sued for bad software or support guarantees a certain quality. Such legal suits in reality are almost non-existent and therefore the guarantees are not so strong at all. The fact that more people can review the code and deliver support for the product seems more important.

Forking
Because the development of a program can happen in a decentralised way the danger exists that at some point, e.g. due to a difference of opinion, different, incompatible versions of the product emerge. This is known as forking. This is a danger but forking is something that in the open source community is perceived as 'not done'. Open source developers see forking as a waste of talent and work because two separate development-processes will exist and the developer community will be split in two. Furthermore the initial developer or initiator of the product could position himself as the governing entity that designates the 'official' versions. It is common use in the open source community that there is one entity (person, group or company) that has control over the project.

Incompatibility
By having access to the source code of a program it is possible, and in fact encouraged, to modify the code to adapt the program to the users specific needs. When these modifications are only meant for internal usage, they are usually not incorporated back into the base version. This leads to backward compatibility problems with the release of new versions of the program. This disadvantage is the main reason for the slow acceptance of the new version (2.0) of the Apache webserver. Many companies modified their 1.x version and are now facing compatibility problems and have to re-implement all these modifications in the 2.0 version.

Giving away precious property
Making software open source is often looked at as giving away a company's most precious resources. Most commercial companies do all that is possible to protect their software's code from outside use. But the source code in itself often does not make it a commercially valuable product. It takes a lot of effort in the area of marketing, creating a well-known name for your company etc. to make your software profitable. However, if you discovered something revolutionary (which only seldom is the case) and want to make money of it, you do not want to release this as open source, but perhaps to file a patent for this software. Filing a software patent explicitly makes the software closed source.

5.3 When to open your source

Deciding whether to go open source or not involves weighing the advantages and disadvantages mentioned above.

The most important reason **not** to make your software open source is because you want to make money from (licensing) your software. Almost every other reason to make or keep your software closed can be lead back to making money. If your code contains some revolutionary idea or technique then it only makes sense not to share it with the world if you want to make money from it.

Most software however, is not meant to make money from. Most software is created because one or the company one works for has a need for this particular piece of software. Here are some gains that a person or company hope to reap by making the software open source:

- product quality and reliability: you have many people reviewing and bug fixing your code
- product maintenance: other people could help you with engineering new software versions and supporting the current versions
- product evolution: create new products around your current product
- third-party recruitment: you want to interest other parties in developing, engineering or supporting your product
- servicing: you want to make money by offering services with the product, like selling consultancy, manuals, distributions, etc.
- you invented something revolutionary and want to share it with the world
- you do not want to support your product.

These are some incentives for making your software open source.

Moving your existing software to open source does not have to take a lot of effort. The easiest way is to attach an open source license and make the source code available on a website for public download. But generally you also want to review your own code and documentation to make the software better understandable.

A next logical step could be to elaborate the software further using the open source community. But making use of this open source development methodology takes a considerable amount of additional costs in terms of money, time and effort. These issues will be further discussed in a later chapter.

6. The Bazaar Development Method

Open source projects have their own development method. This method is often called ‘Bazaar-style’. This method was first formalised by Eric Steven Raymond of the Open Source Initiative in his book called ‘The Cathedral and the Bazaar’. In this book he describes the difference between traditional, commercial development (cathedral-style, a team of specialists creating software with nothing being released before it is ready) and open source development (bazaar-style, everyone doing something his own way with his own ideas, releasing very often). Raymond bases his method on the way that Linux was created. He tested it himself with a small tool he modified using a bazaar-style approach.

Typical bazaar-style development means that there is a large, decentralized (people from over the whole world contribute) community of contributors that have joined the project mainly because of their own interest in it. A programmer is most motivated when the project or part of it solves a problem for him. Project structure is fairly flat: typically one to three project leaders and the rest are (equal) contributors. The project leader delegates all the work and decides which improvements to accept and what and when to release. New releases happen very often, sometimes more than once a day. With this style of development it is very hard to start up a project, because the development community needs something to test, debug and play with. It is therefore that a bazaar-style project almost always starts with a program of some form. When a project leader loses time or interest he has to hand it over to someone else.

6.1 Starting bazaar-style development

When you want to start a project in bazaar-style, the first thing you need is a developer community. In order to come to a large enough community, you need to get people interested. You need what Eric Steven Raymond calls a ‘plausible promise’. This is a program that could be unstable, poorly functioning and undocumented but which can run and which provides the potential to become a really nice and usable program.

The project needs a leader or leaders. The leader needs strong communication skills, as it is he who needs to attract people and interest them in the project. He must delegate the work and keep the contributors happy with the work they are doing. Furthermore the leader needs to have good design and programming skills. Because he decides which improvements will become ‘official’ and which not he needs to be able to recognise a good design and a good piece of code.

Last but not least it will be obvious that the contributors to the project need to be people with at least average design and programming skills. They need to be very motivated to participate in the project.

6.1.1 Open source software development websites

In order to facilitate the simultaneous development of an open source project by a large community the project needs a central place to store the source code, to facilitate the communication between the project members etc. Usually it is not

feasible for an individual project leader to host such a development environment. But the open source community has access to so called open source software development websites. The most well known and largest of these sites is SourceForge.net (This site can be found at http://sourceforge.net).

Features that typically are provided are:

- Project web server
  Contains the homepage for the project with optional additional information
- Code Versioning System
  Enables check-in check-out mechanism for source code files for different versions
- Bug tracking tools
  Tools to facilitate the tracking and reporting of bugs in a project
- Mailing list, discussion forum
  Provide the communications infrastructure that can be used by the project members
- Compile farm
  Provide online access to different computing platforms for compiling and testing the code.

Access to these development sites is easy. They typically require you to become a member of the site. Becoming a member is easy; you just register with a name and some additional information, and access is granted. If you want to start a new project you fill in a form, and usually within one day the project is accepted and an empty project site is created. At the same time all the other facilities are also available to the project. Acceptance is normally a formality given that your project is an open source project.

6.2 Advantages of bazaar-style development

Many of the advantages of developing in bazaar-style can be lead back to the large number of contributors to the project (assuming you managed to attract a lot of people to the project). There is a lot of opportunity for peer-review. It should be obvious that when many people run the program and look at its source code, the chance of finding bugs is greater than when only a few people look at the program. It should also be obvious that when there is a problem the chance it will be solved will be greater. So the large community that is inherent to bazaar-style will lead to a more stable, less buggy product.

Another good thing is that the testers of your code are usually developers too, so there’s code-awareness among your testers. Because of the code-awareness you should get better feedback on your bugs and it should be easier to find them and solve them.

Because the contributors to your project are usually self-selected, the motivation to do work on the project is high. It is the responsibility of the leader of the project to make sure that the motivation stays high. One thing that contributes to this is the fact that there will be many releases. This way, someone who contributed an improvement to the project will not have to wait very long to be rewarded.

A bazaar-style project typically takes place in a decentralized manner. People from all over the world contribute and communication takes place via the internet (project

website, email, forums, etc.). Because of the decentralized nature the costs on resources are lower than in conventional projects, because the people largely take care of their own resources like computers and a work-place.

6.3 Problems with bazaar-style development

After reading articles or books like 'the Cathedral and the Bazaar' one could easily get the impression that bazaar-style development is the solution to every project management problem. This (of course) is not the case. There are some sides and issues to bazaar-style that could very well lead to the fact that a project does not run so well.

Many advantages of developing bazaar-style can be attributed to the fact that the project has a lot of contributors. It could very well be that one cannot find or attract enough people for the project to be successful. This holds especially true during the initial stages of a project. Developers tend to participate in a project if they see a use for themselves for the software being developed in the project. Only when a project has matured enough, so the developers find the program useful, they are willing to participate in the project. It takes persistence from the project lead to get the project to this critical mass. If you are unable to find enough contributors to your project many of the advantages of bazaar-style development vanish.

When a bazaar-style project is very successful (i.e. many contributors, many improvements) it is not unthinkable that a project leader could get overloaded with questions, bugs, fixes etc. resulting in a burn-out. The leader will lose interest in the project or will simply not be able to lead the project anymore which could endanger the whole project.

If for some reason the project must be speeded up this could lead to problems. It simply is very hard to manage people that are placed all around the world and where there is no hierarchical relation between the lead developer and the other developers. Hurrying them up could result in a lot of them losing interest. This is of course a killer for the project. For this reason it is very hard to manage such a project on deadlines.

6.4 Software and platforms

As stated before the open source community is a development oriented community. All criteria a OSI license has to adhere to are there to protect a vision how to create software. There is no mentioning about the use of a specific platform or a specific programming language. So this does not mean that to be open source one must implement the software in Java on the Linux platform.

Of course there can be a difference between the ideology and the practice, meaning that the focus of the community would in effect shift to Java development on Linux. But in practice, at this point this is not the case. The following two tables illustrate this. The numbers in the tables come from SourceForge.net, the world's largest open source software development web site. The two tables list the number of projects by operating system and by programming language.

Operating system	Number of projects
Linux	13664

OS Independent	12287
Microsoft Windows	10620
BSD	1790
MacOS	1523
SunOS/Solaris	1151

Table 1 Number of projects per operating system

As you can see from the table, most open source development is aimed at the Linux platform but the difference with the windows platform is not as big as one would expect when reading about open source. The relative difference becomes even smaller if one takes the OS independent projects into account. These are almost all targeted at both Linux and Windows.

Programming language	Number of projects
C/C++	18294
Java	7046
PHP	5367
Perl	4045
Python	2002
Visual Basic	1081

Table 2 Number of projects per language

This table shows clearly that C/C++ is the most used programming language in the open source community, with the other languages following at a respectable distance. C/C++ is the language of choice for seasoned software developers, so the dominance of C/C++ also underlines the fact that the open source community is a community consisting largely of experienced developers.

As you can see from these tables the perception of open source development being Java development on Linux is just a myth. C/C++ is by far the most used programming language, with the operating system playing not such a significant role.

7. Governments and open source

The importance of software in the society has grown considerably over the last decades. But the downside of this increased importance is the dependency on the software. This should not present a real problem in a world where software is interchangeable and accountable. But in reality there is strong dependency on proprietary software which largely comes from only a few parties. This poses a problem because of the nature of proprietary software. Proprietary software is typically licensed and closed software. Also the control of the future directions of the software development lies in the hands of a few companies in the global market.

This dependency is most visible on the desktop of the average computer. Almost 90% of the desktop computers use one of the versions of Microsoft Windows. Also on the server market based on Intel hardware the Redmond based company has gained a significant market share. For long there was no viable alternative to choose from. But with the advent of Linus Torvalds Linux operating system, now users have an alternative to the Microsoft products.

As Linux is gaining more and more acceptance it becomes a platform of choice for the development of more and more open source applications, tools utilities etc.

These days more governments are starting to encourage the use of open source software, in some cases even making the use mandatory. Their main reason to do so is to make the market more transparent. By giving more alternatives they effectively avoid monopolisation. It decreases the risk on ‘vendor lock-in’; you do not depend on one company for maintaining and fixing your programs. The open source approach has other benefits that fit the responsibilities governments have to the public. By allowing review on the underlying code, programs become more stable and more secure. In this context secure means that it is possible for everyone to review the internal functioning of the program to see if the program functions as expected. It is in the public’s interest to be sure software works as advertised, for instance that particular software doesn’t violate a person’s privacy rights.

Another reason these governments choose for open source software is money. Although this reason is seldom explicitly mentioned, many governments argue that it is not their job to fund privately owned companies with public money. Furthermore because the initial cost of open source software is usually less then when using proprietary software, more money can be spend on computer hardware. For example the Mexican government saved 124 million US dollars by choosing Linux instead of Microsoft Windows, which in turn they spent on extra pc’s for the schools.

Among the countries that invest in open source are Germany, Austria, Spain, India, China and the United Kingdom. Recently the Dutch parliament accepted a motion to instruct the government to actively encourage the usage and the development of open source software in the public sector (Vendrik & Van Tilburg, 2002).

8. Open Source and the TD programme

8.1 Introduction

The Technology Development Programme of the OTEC faculty of the Open University of the Netherlands focuses on Learning Networks (Koper & Sloep, 2003). This programme focuses on the development of new learning technologies in terms of models, prototypical tools and specifications. The results are not only publications in journals, but also software programs.

In the academic world al publications have to meet the following criteria:
1. Accessibility and dissemination
2. Replication
3. Further elaboration of existing work
4. Peer review.

Because software is also considered a publication, it has also to meet these criteria. This means that:
1. the source code and runtime is available for download
2. the source code and documentation provide enough information to rebuild the software
3. a license is attached to the sources and documentation that makes them open source software
4. the software has to be accessible (understandable).

8.2 Software development

For software development in the development programme this means that thought has to be given to the practise of software development. To make the code understandable to peer reviewers there are two paths that can be followed.

One way to create the software is to use the same programming style, language and development environment that was always used. But this implies that the peers asked to review the source code and the documentation needs to have the relevant technological background to be able to understand this programming type.

Another way is to use a programming style more common in the academic world of software development. The most commonly used programming environment in the academic world is Java2 Enterprise Edition (J2EE) using Java as programming language.

The last approach seems to be the most sensible, because this way more peers are available to review the software. Furthermore there are more people able to take on the software and expand it.

Because the others have not been part of the development process they need to be given insight in this process. Furthermore it is essential to give a clear picture of the

structure of the software. So making the software available to others means that even more emphasis has to be given to programming ethics and documentation. Part of this happens automatically because when programming for a public, programmers tend to give more attention to these aspects, because they know that other people will look at their work and they of course do not want to look bad.

Making the code open to further elaboration also means that thought has to be given to the aspect of copyright. It is important to give other people the explicit right to do with the software what you want to allow them to do with it. This is achieved by means of the license added to the software. In the context of the development programme it is necessary to add a license that gives the users the right to study and modify the source code and documentation, whilst still keeping the copyright of the original work. An open source license seems to be the most viable option to achieve this.

Because the programme has little resources it is not possible to create software according to production standards, the results will have the status of prototypes. These prototypes will be delivered in such a way that it is possible to further expand the software. Not only is it not possible to deliver production level software in the timeframes allotted to software development in the programme, it is also explicitly not the intent of the programme to deliver this kind of software tools. The software artefacts are only used to demonstrate concepts of the Learning Networks. Further elaboration of the tools is facilitated, as it is one of the four criteria that have to be met by the software, but it is left to other parties to take up the task.

As stated previously it is not the intent of the development programme itself to take the prototypes further than the scope defined in the project. It is left to third parties to add new functionalities to these programmes or to enhance the quality of them. Therefore the activities in the development programme are not typically those of open source development. Adding a license, changing the development platform and language to a commonly accepted one and focusing more on documentation are merely prerequisites for open source development.

Should one choose to elaborate the prototypes further, the prerequisites for open source development are met, but some additional considerations come to surface. One needs to form a community of developers. Most important for achieving this is a ‘plausible promise’, one needs to present something that appeals to others, something that others are willing to commit to. The community needs a leader that is able to keep people interested in the project, so good communication skills are required. He also needs good design and programming skills, because he has to decide which elaborations get adopted in the software and which do not. If the development programme wishes to follow this path, it has to give thought to these aspects. This means deciding on a strategy to create a community, delivering an interesting idea and choosing a leader.

9. Conclusion

This report described the early beginnings of free software and open source. It explained what free software means and why it is called free software. Free software plus the Netscape Mozilla initiative has eventually evolved to the open source initiative: a less religious, more pragmatic approach to open source development.

This open source approach has advantages like higher reliability, no vendor lock-in, lower prices and ‘openness’ of the code. Of course there are also disadvantages, being no licensing fees, forking and incompatibility. Movement to open source development takes considerable investments and therefore one should carefully analyze if the benefits outweigh these investments.

Open source comes with its own development style, called the ‘Bazaar development style’. A successful bazaar style project has a plausible promise, a large, involved, developer community and a charismatic leader. Its strong points are quality and stability of the product by peer review, higher motivation by the self select mechanism of the community, code awareness of the testers and the low overhead costs. It is however hard to form a community large enough to profit from these benefits, it potentially overloads the project leader and it is virtually impossible to manage the project according to deadlines.

There is a growing interest among governments in open source. Some governments are already embracing open source products and development while others are looking into the possibilities.

As far as software development in the context of the technology development programme is concerned, it is only the intention to create open source software that is ready for open source development. This can be achieved by choosing a common programming environment (J2EE and Java), providing good documentation and adding an open source license. Should the technology development programme wish to take the software one step further, some additional considerations emerge, like building a community and providing a leader.

References

Barr, J. (2001). *Live and let license.* Retrieved 12 November 2002, from LinuxWorld.com http://www.itworld.com/AppDev/350/LWD010523vcontrol4/pfindex.html

Bezroukov, N. (1999). Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism) in *First Monday, 4, number 10.* Retrieved from http://firstmonday.org/issues/issue4_10/bezroukov/index.html

GNU. The *GNU Project.* Boston: Free Software Foundation. [available online at: http://www.gnu.org/gnu/thegnuproject.html].

Hecker, F. (2000). *Setting Up Shop: The Business of Open-Source Software.* Retrieved 19 November 2002, from http://www.hecker.org/writings/setting-up-shop.html.

Koper, E.J.R., & Sloep, P. (2003). *Learning Networks.* Heerlen: Open University of the Netherlands.

Mendys-Kamphorst, E. (2002, September). *Open vs. closed: Some consequences of the open source movement for software markets.* Den Haag: CPB.

Perens, B. (1998). *The Open Source Definition, version 1.9.* Redwoord City, CA: Open Source Initiative. [available online at http://www.opensource.org/docs/definition.php].

Raymond, E.S. (1999). *The Magic Cauldron.* [available online at http://catb.org/~esr/writings/magic-cauldron/magic-cauldron.html].

Raymond, E.S. (2000). *Homesteading the Noosphere.* [available online at http://catb.org/~esr/writings/homesteading/homesteading/].

Raymond, E.S. (2001). *The Cathedral & the Bazaar.* Sebastopol, CA: O'Reilly.

Ritchie, D.M. (1984). *The Evolution of the Unix Time-sharing System.* Murray Hill: Bell Laboratories. [available online at http://cm.bell-labs.com/cm/cs/who/dmr/hist.html].

Vendrik, K., & Van Tilburg, R. (2002). *Software open u!* [available online at http://www.groenlinks.nl/partij/2dekamer/publikaties/SoftwareOpenU!.htm].

Annex I The General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it,

either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

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.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Annex II Relevant websites

Free Software Foundation (FSF)	http://www.fsf.org
GNU	http://www.gnu.org
open source Initiative (OSI)	http://www.opensource.org
The cathedral and the bazaar	http://www.tuxedo.org/~esr/writings/cathedral-bazaar/
SourceForge.net	http://www.sourceforge.net
open source Software Development as a special type of Academic Research	http://www.softpanorama.org/OSS/index.html
Software open u!	http://www.groenlinks.nl/partij/2dekamer/publikaties/SoftwareOpenU!.htm