Open Source

Entstehung, Entwicklung, Einfluss auf die moderne Computergesellschaft

Seminarkursarbeit von

Christopher R. Nerz Michael S. Walz

12. Klasse
Schuljahr 2003/04

Vorwort

Als in unserer Schule, dem Kepler–Gymnasium Tübingen, die Wahlmöglichkeiten für die Oberstufe erläutert wurden, stieß bei uns vor allem der Seminarkurs auf reges Interesse. Nicht zuletzt deshalb, weil man damit ein bisschen den Prüfungsstress des Abiturs abbauen kann. So war die Entscheidung, dass wir am Seminarkurs teilnehmen würden, schnell gefasst, aber das Thema bereitete uns Kopfzerbrechen.

Natürlich wollten wir beide ein Thema, das mit Computern in Verbindung steht, wählen, da unsere Hobbies größtenteils mit dem Computer zu tun haben. Dummerweise muss unser Seminarkurs dem gesellschaftswissenschaftlichen Bereich zugeordnet werden können. So brachte uns unser Lehrer Herr Dr. Bitsch auf die Idee, ein Thema, das in Verbindung mit Open Source steht, zu wählen, da man dort relativ einfach gesellschaftswissenschaftliche Teilthemen finden kann. So wählten wir das Thema **„Open Source: Entstehung, Entwicklung, Einfluss auf die moderne Computergesellschaft“**.

Nachdem schon von Anfang an feststand, womit wir unsere Arbeit verfassen würden, nämlich mit LaTeX, da es selbst einiges mit Open Source zu tun hat, fingen wir an Material zu sammeln. Unsere Hauptquelle stellte logischerweise das Internets dar, da die Geschichte des Internet untrennbar mit Open Source verknüpft ist. Später fiel uns ein, dass bestimmte Begriffe, wie *Source Code* oder *Linux* nicht allen unseren Zuhörern und Lesern bekannt sind, und deshalb findet man die wichtigen Begriffe im Appendix unter A. Außerdem sind die Quellen im Appendix unter B zu finden, die Internetseiten liegen ausgedruckt bei.

Es sei noch kurz erwähnt, dass wir als Programmierer „von Berufswegen“ an Open Source interessiert sind. Dennoch versuchen wir so unvoreingenommen wie möglich über dieses Thema zu referieren.

Inhaltsverzeichnis

1 Open Source Definition

Unter Open Source wird allgemein frei verfügbare, quelloffene Software verstanden. Aber nicht alles, von dem der Source Code vorliegt, darf sich auch „Open Source“ nennen und auch die freie Verfügbarkeit zeichnet ein Programm zunächst höchstens als „Freeware“ aus. Open Source ist mehr:

Der Begriff wurde von der *Open Source Initiative[15]* geschützt und seine Bedeutung wurde genau definiert. Auf die Definition der *Open Source Initiative* wird im Abschnitt *Open Source Lizenzen* eingegangen und hier soll nur kurz dargelegt werden, was im Allgemeinen unter „Open Source“ verstanden wird:

1. Wie die Bezeichnung „Open Source“ schon andeutet, ist der Source Code frei verfügbar.
2. Die Software darf inklusive Source Code frei weitergeben werden.
3. Die Software darf weiterentwickelt werden.
4. Für diese weiterentwickelte Software gelten die gleichen Bedingungen wie für die Ausgangssoftware.

2 Ursprung von Open Source

Eigentlich müsste dieses Kapitel statt „Ursprung von Open Source“ „**Wieder**entstehung von Open Source“ oder so ähnlich heißen, da die momentan zu beobachtende Entwicklung in Richtung des Open Source eigentlich nur ein **Wiederzurück**kehren zu älteren Traditionen ist. Dies sei im Folgenden kurz erläutert:

2.1 Entstehung der kommerziellen Software

Bis etwa 1965 galt es als normal Software als kostenlose Beigabe zu einem neuen Rechner auszuliefern, selbiges traf auch für den Source Codes zu den Programmen zu. Die Hersteller verdienten nur an der Computer–Hardware. Die Source Codes waren für Programmierer in der ganzen Welt frei zugänglich und durften weiterentwickelt werden, es handelte sich also um Open Source Software und die Hersteller optimierten ihre Programme mit Hilfe außenstehender Programmierer. Spätestens in den siebziger Jahren stellten Programmierer dann fest, dass sich mit ihren Programmen gute und solide Einkünfte erzielen ließen. Es wurde begonnen die Weitergabe von Programmen mit Hilfe von Lizenzen einzuschränken oder gar ganz zu verbieten um so diese Einnahmequellen offen zu halten. Die kommerzielle Software war geboren und der Source Code wurde zum best gehüteten Geheimnis des jungen IT–Marktes. Bereits zehn Jahre später gab es kaum noch aktuellen frei verfügbaren Source Code. Verschwiegenheitsvereinbarungen, so genannte Non–Disclosure–Agreements, hinderten Programmierer an der freien Weiterentwicklung ihrer Produkte. Seit dem waren Computer–Anwender bei Programmierfehlern oder Sonderwünschen ganz auf das Entgegenkommen der Software–Firmen angewiesen.

2.2 Ursprung des heutigen Open Source

Richard Stallman vom Massachusetts Institute of Technology (MIT) war 1984 mit dieser Entwicklung so unzufrieden, dass er beschloss ein wieder freies Programmpaket zu entwickeln, er nannte es GNU. Zusammen mit dem von dem finnischen Studenten Linus Torvalds entwickelten Unix ähnlichem Kernel, Linux, bildet GNU das vollständige Betriebssystem GNU/Linux. Ziel von Stallman war es, die freie und offene Zusammenarbeit der Software–Programmierer, wie er sie zu Beginn der siebziger Jahre selbst noch erlebt hatte, wieder zu ermöglichen – zum Nutzen aller Computer Benutzer. Nach Ansicht des MIT Computer–Experten müssen alle Source Codes vervielfältigt, verändert und weitergegeben werden können. Laut ihm ist Software nur dann frei, wenn sie für jeden uneingeschränkt nutzbar ist. Mit dieser Überzeugung gründete er und seine Mitstreiter 1985 die Free Software Foundation (FSF), die diese Überzeugungen bis heute kompromisslos vertritt. Um die Freiheit der Software zu schützen wurde die GNU General Public License geschaffen. Es war dann der Software–Experte Eric S. Raymond, der 1998 vorschlug Software mit offenem Source Code in Zukunft als „Open Source Software“ zu bezeichnen.

3 Die Entwicklung eines Open Source Programms

3.1 Die allgemeine Entstehung von Software

Im Allgemeinen entsteht Software durch einen mehr oder weniger langwierigen Wechsel aus Testen und Programmieren. Nachdem die Grundidee für ein Programm oder einen Programmteil aufgekommen ist, wird im Normalfall zunächst das Programm auf theoretische Weise durchgegangen, so dass dabei eventuelle Probleme, Strukturschwierigkeiten und grundlegende Definitionen geklärt werden können. Anschließend wird überprüft, ob es sich lohnt ein solches Programm zu schreiben, das heißt ob der Nutzen in einer geeigneten Relation zu den aufkommenden Kosten, beziehungsweise dem Arbeitsaufwand steht. Kommt man dabei zu einem positiven Ergebniss, kann begonnen werden das theoretische Programmier Konzept zu entwicklen.

Dabei wird die große Gesamtaufgabe in mehrere kleinere Probleme aufgeschlüsselt, die dann, je nach Personalmöglichkeiten, auf mehr oder weniger Personen verteilt werden. Bevor jetzt die eigentliche Programmierung beginnen kann, wird jeder Teil des Programms skizziert und damit die grundsätzliche Basis des Programmes definiert. Dabei einigen die Programmierer sich im Idealfall auf gemeinsame Standards, wie zum Beispiel eine geeignete Konvention.

Erst jetzt geht die eigentliche Programmierung los. Zunächst wird nur ein Grundprogramm geschrieben, es enthält meistens viele Fehler, welche oft als Bugs bezeichnet werden. Sobald dieses Grundprogramm so weit funktionsfähig ist, es wird nun meist als alpha–Version bezeichnet, wird die erste Testreihe gestartet. Die dabei auftauchende Probleme und Fehler werden den Programmierern mitgeteilt. Diese versuchen daraufhin Patches, Erweiterungen und Verbesserungen des Programms zu schreiben und so nach und nach die Probleme zu lösen. Ist das Programm stabil genug, kann es, je nach Personal und sonstigen Möglichkeiten, noch eine zweite Testreihe gestartet, Beta–Test genannt. Nach und nach werden so, wenn genug Zeit und sonstige Ressourcen zur Verfügung stehen, alle oder zumindest die meisten Fehler gelöst. Nun kann das Programm um verständliche Bedienungsvereinfachungen und anderen „Komfort“ erweitert werden, um anschließend veröffentlicht zu werden. Kommt es trotz der Test zu weiteren Problemen und Fehlern, folgen nach Veröffentlichung der Software noch weitere Patches. Die Software wird somit noch im Nachhinein verbessert.

Im Folgenden will ich darauf eingehen, inwieweit Open Source Programme sich bei dieser Entwicklung von kommerzieller Software unterscheiden:

3.2 Bei kommerzieller Software

Bei den meisten kommerziellen Software–Entwicklungen gibt es vor allem zwei begrenzende Faktoren: Zeit und Geld. Das heißt, dass die Test nicht ausgiebig genug sind, um alle Fehler zu finden. Somit kommt eine Software heraus, die viele Fehler und dadurch meist auch Sicherheitslücken enthält. Desweiteren stellt sich bei kommerzieller Software noch ein weiteres Problem: Auf Grund der meist hierarchischen Ordnung verkompliziert sich die Fehlerbehebung. Das sei nochmals erklärt: Bei Unternehmen sind oft die Programmierer und die Personen, die Fehlerbeschreibungen entgegen nehmen, nicht in einer „Gruppe“, daraus folgt, dass Fehlerbeschreibungen erst weitergeleitet werden müssen (meist zunächst an Vorgesetze und deren Vorgesetzte) bis sie schließlich die Programmierer (direkt oder indirekt) erreicht werden. Dieser Vorgang kostet wieder Zeit, somit können wieder weniger Fehler gelöst werden, da ja nur begrenzt Zeit zur Verfügung steht.

Ein weiterer Punkt ist, dass die Relation von Nutzen und Kosten meist rein finanzieller Betrachtung entspringen, da mit kommerziellen Programmen meist hauptsächlich ein möglichst großer Gewinn erreicht werden soll.

3.3 Bei Open Source Software

Was bei den meisten Open Source Projekten, vor allem solang sie noch unbekannt sind der beschränkende Faktor ist, sind Programmierer. Meist kommt zunächst eine Person auf die grundsätzliche Idee zu einer neuen Software. Diese wird dann meist in einem Forum oder auch mehreren Foren beschrieben und um (je nach Komplexität) viele Programmierer und sonstige Mitarbeiter geworben. Können genug Mitarbeiter gewonnen werden und sind auch die nötigen Programmier– und sonstigen Kompetenzen vorhanden, kann nun der oben beschriebene Prozess beginnen.

Da Open Source Programme nicht finanzielle Ziele haben, wird die Relation von Nutzen und Kosten anders „berechnet“ als bei den kommerziellen Gegenstücken. Nutzen meint bei Open Source wirklich Nutzen, das heißt in wie weit dieses Programm für die Open Source Szene oder für andere mögliche Anwender von Nutzen sein kann. Kosten meint hier Programmieraufwand und Zeit der Beteiligten.

Auch das Problem der komplizierten „Fehlerbeschreibungs–Weitergabe“ stellt sich bei Open Source meistens nicht, da Fehler fast immer öffentlich in Foren vorgestellt werden. Auf diese Weise haben die Programmierer auch die Möglichkeit, der Person, die einen Fehler beschreibt, direkt Fragen zu stellen. Da Tester nicht angestellt werden müssen, sondern auf freiwilliger Basis gesucht werden, kann meist eine sehr große Testergemeinde gefunden werden, die sozusagen automatisch mit der Komplexität und Qualität des Programmes zunimmt.

Doch auch bei Open Source Projekten gibt es ein immer wieder aufkommendes Problem, das vorallem sehr große Projekte während der Anfangsphasen und kleine Projekte nach einer Weile bekommen: Wird ein Projekt uninteressant, läuft nicht gut oder es können nicht genug Personen begeistert werden, kann es zum „Aussterben“ von Projekten kommen.

3.4 Verbreitung

Unter anderem folgt aus diesen meist besseren Voraussetzungen die wachsende Verbreitung von Open Source. So besagt eine Umfrage der bekannten Computerfachzeitschrift iX[22], dass in 74% der Unternehmen in irgendeiner Form Linux eingesetzt wird und diese Zahl langsam noch ansteigt. Dagegen nimmt die Verbreitung kommerzieller Unix–Varianten von beispielsweise IBM und Sun Microsystems stetig ab. Heutzutage ist das Internet ohne Linux kaum noch denkbar. Laut Untersuchungen lief auf 44% der 2001 in Deutschland mit dem Internet verbundenen Rechner Linux. Nur auf etwa 30% ein Betriebssystem von Microsoft, sprich ein Windows-System.[1] Aber Open Source Software ist noch mehr als nur Linux. Beispielsweise ist der Web Server Apache fester Bestandteil des World–Wide–Web, dessen Marktanteil beträgt etwa 60%.

[1] Zu beachten ist, dass damit wirklich alle in Deutschland mit dem Internet verbundene Computer gemeint sind, nicht nur Privat–PCs, sondern zum Beispiel auch Internet–Server und Router

4 Open Source Lizenzen

4.1 Definition

Wer bestimmt, was eine Open Source Lizenz ist, und was nicht?

Kann jeder eine Lizenz veröffentlichen und diese als Open Source Lizenz bezeichnen?

Nein, natürlich nicht. Es wurde von der Open Source Initiative eine klare Definition[16] ausgearbeitet, wie eine Open Source Lizenz auszusehen hat:

1. **Freie Weitergabe:** Die Lizenz darf niemanden in seinem Recht einschränken, die Software kompiliert oder als Quellcode in irgendeiner Form zu verschenken oder zu verkaufen, ebenso wenig darf sie irgendwelche Gebühren für die Weitergabe festschreiben.

2. **Quellcode:** Der Quellcode muss dem Programm beigelegt werden, falls dem nicht so ist, muss es eine allgemein bekannte Möglichkeit geben, den Quellcode zum Selbstkostenpreis zu bekommen, vorzugsweise als gebührenfreier Download aus dem Internet. Auch muss der Quellcode eine allgemein lesbare Form haben, absichtlich unverständlich geschrieben oder nachträglich gemachter Code ist daher unzulässig.

3. **Abgeleitete Software:** Die Lizenz muss Veränderungen zulassen, außerdem muss sie es zulassen, dass die solcherart entstandenen Programme unter denselben Lizenzbestimmungen weitervertrieben werden können wie die Ausgangssoftware.

4. **Unversehrtheit des Quellcodes:** Die Lizenz darf nur die Veränderung des Quellcodes nur dann einschränken, wenn sie die Möglichkeit gibt, den Quellcode durch „Patch files“ weiter zu verändern und die Weitergabe von Software, die aus veränderten Code entstanden ist ausdrücklich erlaubt. Die Lizenz darf verlangen, dass abgeleitete Programme einen anderen Namen oder eine andere Versionsnummer als die Original–Software tragen.

5. **Diskriminierung:** Die Lizenz darf niemanden, weder einzelne Personen noch Personen–Gruppen, benachteiligen.

6. **Einschränkungen des Einsatzfeldes:** Die Lizenz darf niemanden daran hindern, das Programm in einem bestimmten Bereich, wie z.B. der Genforschung, einzusetzen.

7. **Weitergabe der Lizenz:** Die Rechte an einem Programm müssen auf alle Personen übergehen, die diese Software erhalten, ohne dass sie eine eigene Lizenz erwerben müssen.

Jede Lizenz, die diese Bestimmung erfüllt, ist eine Open Source Lizenz. Demzufolge gibt es sehr viele Open Source Lizenzen. Zum Beispiel fallen folgende Lizenzen darunter:

- Apache–Lizenz
- Apple Public Source License
- BSD License
- GNU General Public License (GPL)
- GNU Lesser General Public License (LGPL)
- IBM Public License
- Intel Open Source License
- Mozilla Public License

Hier soll im Folgenden vor allem auf die *GNU General Public License[7]* und die *GNU Lesser General Public License[8]* eingegangen, beide herausgegeben von der *Free Software Foundation[4]*. Im Anschluss soll noch die *GNU Free Documentation License[6]* behandelt werden, die zwar keine Open Source Lizenz im obigen Sinne darstellt, da sie sich nicht auf Software sondern auf Schriftstücke bezieht. Allerdings sind fast alle Schriftstücke, die zu Open Source gehören unter dieser Lizenz veröffentlicht.

Im Folgenden werden die Grundgedanken der jeweiligen Lizenz in eigenen Worten wiedergegeben:

4.2 GNU General Public License (GPL)

Im Normalfall gibt es eine Lizenz, um dem Benutzer zu verbieten, die Software zu vervielfältigen oder zu verändern. Im Gegensatz dazu soll die *GPL* garantieren, dass jeder die Software benutzen, kopieren, weitergeben und verändern darf bzw. kann.
So sichert die Lizenz die Freiheit,

1. die Software für jeden Zweck nutzen zu dürfen.
2. die Software kostenlos oder gegen eine Gebühr weiterzugeben. Der Quellcode **muss** der Software beiliegen oder öffentlich zugänglich sein. (Zum Beispiel per kostenlosem Download).
3. den Quellcode zu verändern und so die Software den eigenen Bedürfnissen entsprechend anzupassen.
4. die von Ihnen veränderte Software, wie in Punkt 2 beschrieben, weiterzugeben.

Die *GPL* stellt die *dauerhafte* Freiheit der Software dadurch sicher, dass jede Software, die von dieser *abgeleitet* ist, das heißt Teile von ihr benutzt, wieder unter die *GPL* gestellt werden muss.

So ist es ohne Bedeutung, ob sie die Software aus erster oder zum Beispiel zehnter Hand erhalten haben. Da Sie die Software nur unter gleichen Bedingungen weitergeben dürfen, wie Sie sie bekommen haben, können auf den „Zwischenstationen“ keine Rechte „verloren“ gegangen sein.

Wenn der Quellcode verändert wurde, muss durch einen Kommentar dokumentiert werden, wer welche Änderung wann gemacht hat, sodass immer ersichtlich ist, vom wem bestimmte Teile des Programms wirklich entwickelt wurden.

Allerdings verbietet die *GPL* die Einbindung in kommerzielle Produkte. Das heißt, eine Library, die unter dieser Lizenz verbreitet wird, kann nicht zur Erstellung von kommerziellen Produkten benutzt werden.

4.3 GNU Lesser General Public License (LGPL)

In den Grundgedanken stimmt die *Lesser GPL* mit der *GPL* überein. Es gibt lediglich einen wichtigen Unterschied:

Die *Lesser GPL* erlaubt die Einbindung in kommerzielle Produkte. Das ist der Grund, weshalb die Lizenz früher *GNU Library General Public License* hieß. Allerdings führte das dazu, dass viele Libraries unter diese Lizenz gestellt wurden, obwohl die *GPL* besser gewesen wäre.

Man sollte eine Library nur unter der *Lesser GPL* veröffentlichen, wenn es dazu Alternativen für kommerzielle Benutzung gibt. Gehen wir davon aus, die Library wäre einmalig, dann verschafft man den anderen Entwicklern „freier Software“ mehr Vorteil, wenn man sie unter der *GPL* veröffentlicht. Wenn es allerdings Alternative gibt, sieht das anders aus: Die kommerziellen Entwickler dürfen die Open Source Library nicht benutzen, also nehmen sie einfach eine andere. Daher bringt es dann keinen Nutzen, die Library auf die „freie Software“ zu beschränken, stattdessen sollte man die Library unter der *Lesser GPL* veröffentlichen. Vielleicht wird sie dann von kommerziellen Entwicklern benutzt und vielleicht wird sie sogar die Standard–Library auf diesem Gebiet.

4.4 GNU Free Documentation License (FDL)

Diese Dokumentationslizenz ist geschaffen worden, um die Dokumente,[2], die zu den unter der *GPL* veröffentlichten Softwareprodukten gehören, unter eine ähnliche Lizenz zu stellen wie die Software selbst.

In der Präambel der *FDL[6]* steht unter anderem:

> „Freie Software braucht freie Dokumentation."

Dies leuchtet ein, wenn man bedenkt, dass

1. die Dokumentation zu der Software jedem genauso zugänglich sein sollte, wie die Software selbst.

2. wenn die Software verändert wurde, es erlaubt sein muss, das Handbuch anzupassen.

Natürlich ist die *FDL* nicht auf Softwarehandbücher beschränkt. Die *Free Software Foundation[4]* empfiehlt prinzipiell die Verwendung dieser Lizenz für Werke, die als Anleitungen oder Referenzen dienen sollen.

4.5 Vergleich: GPL u. Windows XP EULA

Ursprünglich sollte hier ein Vergleich der GNU General Public License (GPL) und der Microsoft Windows XP EULA (EULA) stehen, wobei die beiden Lizenzen inhaltlich untersucht und ausführlich verglichen werden sollten. Dies scheiterte aber daran, dass die beiden Lizenzen inhaltlich zu verschieden sind.

Deshalb sollen hier nun „nur" die Grundgedanken der EULA aufgelistet und anschließend die wenigen Gemeinsamkeiten der Lizenzen beschrieben.

4.5.1 Grundgedanken der EULA

Die Microsoft Windows XP EULA erlaubt dem Benutzer das Produkt (in diesem Fall also Windows XP) auf maximal einem Rechner zu installieren, zu benutzen, anzuzeigen und auszuführen. Dieser eine Rechner darf maximal zwei (2) Prozessoren haben.
Die EULA verbietet dem Benutzer eigentlich alles Erdenkliche. Zum Beispiel verbietet sie:

- mehr als 10 Verbindung zu dem Rechner aufzubauen.

- andere Remote–Controll–Tools (Bsp.: WinVNC) als die von Microsoft mitgelieferten zu benutzen.

- das Produkt zu verleihen. Verkauft werden darf es nur einmal (1x) vom ursprünglichen Besitzer. (Ein Verkaufen an einen Freund ist legal. Das Zurückkaufen allerdings nicht.)

- ...

[2] wie zum Beispiel Handbücher, Referenzen etc.

Außerdem erlaubt die EULA noch viele Aktionen seitens Microsoft. Zum Beispiel erlaubt sie:

- dass eine Hardwareänderung oder Neuinstallation eine erneute Aktivierung des Produktes nach sich zieht
- dass Microsoft technische Daten über die Benutzung sammelt und diese Dritte zu Verfügung stellt. Die Daten dürfen alles enthalten, nur nichts, was einen *direkt* identifiziert. (Hier stellt sich die Frage, ob z.B. eine E–Mail–Adresse einen *direkt* identifiziert oder nur *in*direkt über den Provider.)
- dass Microsoft *automatisch* Updates auf dem System einspielt.
- dass, sobald urheberrechtlich geschützte Daten wie Musik, Video etc. auf dem Rechner liegen, Microsoft Kontrollprogramme installieren darf, die ein Verschieben, Kopieren, Betrachten oder Backup anfertigen verhindern.
- ...

Des weiteren weist Microsoft jegliche Gewährleistung und Verantwortung von sich. Das absolute Maximum, das man bekommen könnte, wenn der absolute Schlimmstfall (Systemabsturz, alle Daten vernichtet, keine Neuinstallation möglich, ...) eintritt ist, dass man den Preis, den man für das Produkt gezahlt hat, zurückbekommt. (Oder 5 US $, falls man weniger gezahlt hat) Dies gilt aber nur innerhalb der ersten 30 bzw. 90 Tage.

Spekulation: Also wäre es theoretisch möglich, dass Microsoft *automatisch* ein Update nach z.B: 100 Tagen einspielt, mit dem die gesamte Software nicht mehr funktioniert. Dann hätte man *keinerlei* rechtliche Ansprüche an Microsoft.

4.5.2 Gemeinsamkeiten: GPL u. Windows XP EULA

So grundverschieden die beiden Lizenzen auch sind, es gibt doch ein paar Gemeinsamkeiten:

1. Beide Lizenzen weisen jegliche Garantieansprüche von sich. Das Benutzen der unter den Lizenzen stehenden Produkte geschieht auf eigene Gefahr. Bei der GPL muss allerdings gesagt werden, dass dadurch, dass das Produkt frei verfügbar ist, der oder die Entwickler überhaupt nicht das Geld mit dem Produkt verdienen, um Garantieansprüche bezahlen zu können.
2. Beide Lizenzen stellen klar, dass jemand der die Lizenz „erworben“ hat, damit nicht Besitzer des jeweiligen Produktes ist. Man darf das Produkt nur — mit den von der Lizenz zugesprochenen Rechte — benutzen. Die GPL räumt einem aber erheblich mehr Rechte als die EULA ein.
3. Beide Lizenzen verfallen, wenn man eine Aktion ausführt, die laut ihr nicht erlaubt ist. Man verliert dadurch sämtliche Nutzungsrechte, bekommt aber das Geld, das man für den Erwerb der Lizenz bezahlt hat, nicht zurück.

5 Vor/Nachteile

Wie schon aus der Überschrift ersichtlich ist, geht es in diesem Abschnitt um die Vor– und Nachteile von Open Source. Die wichtigsten sollen im Folgenden aufgeführt und erläutert.

5.1 Vorteile

kostengünstig

Open Source ist kostengünstig. Die Software kann zum Selbstkostenpreis erworben werden. Der Hersteller ist zwar nicht verpflichtet, die Software günstig bereitzustellen, aber es gibt ja die Möglichkeit sie von Freunden zu kopieren, oder aus dem Internet zu laden. Und bei Open Source ist das legal[3]. Selbiges gilt für fehlerbehebende Nachbesserungen oder Erweiterungen des Original–Programmes, welche bei mancher proprietären Software fast soviel wie das Original–Programm kostet.

wenig Marktdruck

Von einer Open Source Software weiß der Markt meistens erst, wenn die erste funktionsfähige Version fertiggestellt ist. Davor ist das Projekt oft nur einer kleinen Gruppe interessierter Personen bekannt. Das heißt, dass es nicht den Druck gibt, den ein kommerzielles Projekt erfährt, das zum einem von oben[4] einen Fertigstellungstermin genannt bekommen hat, und zum anderen in der Presse schon fast belächelt wird, weil dieser Termin schon öfters verschoben wurde. Der häufigste Fertigstellungstermin für ein Open Source Projekt ist: „It's done when it's done!“ Außerdem kann ein Open Source Projekt es sich meistens nicht leisten, große Werbekampagnen zu starten. Bei kommerziellen Produkten führt dies manchmal dazu, dass die Software unbedingt zu einem bestimmten Zeitpunkt auf den Markt muss, weil sonst der Werbeeffekt verfällt und das Geld, das man für ihn eingesetzt hat, verloren gehen würde.

Durch den geringeren Druck können sich die Entwickler mehr Zeit lassen, was zweifelsohne die Qualität des Produktes verbessert. (s. Sicherheit 1)

[3] Niemand braucht ein schlechtes Gewissen haben, weil er die Software nicht beim Hersteller gekauft, sondern von einem Freund kopiert hat.

[4] d.h. vom Vorgesetzten, dem Vorstand oder ähnlichen

Sicherheit

Open Source ist häufig sicherer als Software, deren Quellcode nicht frei verfügbar ist. Dies liegt vor allen an den ersten zwei der folgenden drei Gründen:

1. Sicherheitslücken werden ja nicht absichtlich in ein Programm integriert[5], sondern entstehen unabsichtlich, durch Programmierfehler, weil man etwas übersehen hat, oder weil der Druck das Produkt schnell auf den Markt zu bringen zu groß ist (s.o.). Frei nach dem Motto, dass viele Augen mehr sehen als nur zwei, können bei Open Source Software Fehler und Sicherheitslücken von vielen Entwicklern in aller Welt schnell aufgespürt und behoben werden. Auf diese Weise wird bei vielen Open Source Programmen eine hohe Stabilität, Effizienz und eine gute Fehlerfreiheit erreicht.
2. Da der Quellcode offen gelegt werden muss, können die Entwickler keine versteckten Hintertüren einbauen, weil dies ja sofort öffentlich würde. Zum Beispiel wurde am 04.06.2004 von einer Hintertür bei Netgears Access–Point WG602v1 berichtet[9], über die man vollen Zugriff auf den Access–Point erlangen kann. Anscheinend lautet der Anmeldename „super“, während das Passwort die Telefonnummer des Herstellers z–com in Taiwan sei. Drei Tage später war zu lesen[9], dass die Hintertür nicht geschlossen, sondern in der neuen Version lediglich die Anmeldedaten geändert wurden. Weiter zwei Tage danach wird gemeldet, dass die Hintertür „endgültig geschlossen sein soll.“ Wie aus der Formulierung entnommen werden kann, weiß man nicht, ob sie auch wirklich „geschlossen worden ist.“ Ohne den Quellcode lässt sich das nicht ohne größeren Aufwand überprüfen.
3. Anscheinend wollen ein Teil der Virenentwickler (bei weitem nicht alle) mit ihren Viren keinen bzw. kaum Schaden anrichten, sondern lediglich erreichen, dass die Sicherheitslücken geschlossen werden. Ihrer Ansicht nach haben sie fast keine andere Möglichkeit dazu, denn die wenigsten Firmen würden Sicherheitslücken entfernen, wenn sie der Firma zwar bekannt wären, die Öffentlichkeit aber nichts über sie wüsste. Das würde nur unnötig Geld kosten. Bei Open Source kann der jeweilige Virenentwickler den Fehler selber beseitigen und dem ursprünglichen Hersteller zusenden. Dieser wird den verbesserten Code eigentlich immer einbauen, weil es für ihn ja fast keine Arbeit bedeutet, aber seine Software trotzdem verbessert.

[5] sollte man zumindest hoffen

Patches erscheinen schneller

Fehlerbehebende Patche, so genannte Bug–Fixes, für Programme, bei denen Fehler beobachtet wurden, sind bei Open Source Entwicklungen meist schneller öffentlich verfügbar. Das liegt daran, dass bei den meisten Firmen Fehler bei einer *„Beschwerden–Stelle“* eingereicht werden, um dort auf ihre Richtigkeit überprüft zu werden. Erst danach wird diese Fehlerbeschreibung die hierarchische Leiter noch oben weitergeleitet, bis sie eine Person erreichen, welche genug Entscheidungskompetenz hat, die Programmierung des Bug–Fixes anzuordnen. Dann geht die Anordnung den Bug–Fix zu schreiben die hierarchische Leiter wieder nach unten, bis sie eine Person mit der benötigten Programmiertechnischen Kompetenz erreicht. Erst jetzt beginnt die Arbeit am eigentlichen Patch. Ist diese dann endlich abgeschlossen läuft das Ganze nochmals rückwärts ab, bis der Bug–Fix schließlich veröffentlicht wird.
Im Gegensatz dazu muss der Fehler bei Open Source Programmen „nur“ *öffentlich* gemeldet werden, zum Beispiel in einem dem entsprechenden Forum. Programmierer die sich mit diesem Programm beschäftigen lesen ihn, können sich dann *direkt* an die Programmierung des Bug–Fixes und dessen *(wieder direkt)* anschließende Veröffentlichung machen, damit entfällt das Auf– und Absteigen einer hierarchischen Leiter. Ein weiterer Faktor ist, dass viele Open Source Gruppen so genannte Pre–Patches herausbringen, das heißt Patches die zwar diesen Fehler beheben, aber nicht fertig getestet wurden. Somit ist es Benutzern möglich zu entscheiden, ob er sozusagen als „Patch–Tester“ fungieren will und so den Patch früher erhält, oder ob er wie gewöhnlich wartet, bis der Patch ausreichend getestet wurde. Dieses Vorgehen ist einem Konzern unmöglich, da von ihm prinzipiell erwartet wird „fertige“ Patches abzuliefern.

Anpassung an eigene Bedürfnisse möglich

Es lassen sich leicht Änderungen an der Software vornehmen, sodass es den eigenen Bedürfnissen besser entspricht. Man muss „nur“ den frei verfügbaren Quellcode erweitern bzw. verändern und danach die Software neu kompilieren und linken. Dies ist für Privatpersonen nicht so interessant, da es das entsprechende Know–How erfordert. Für Firmen und Institutionen ist es das aber sehr wohl. Diese können Aufträge vergeben, um die entsprechende Software erweitern lassen. Da die Entwickler nicht von Null anfangen müssen, ist diese Methode um einiges billiger als eine komplette Software neu schreiben zu lassen.

viele, gute Übersetzungen

An fast jeden Open Source Projekt arbeiten Entwickler verschiedener Nationalitäten. Da jeder dieser Entwickler die Software gerne in seiner eigenen Sprache hat[6], besitzen die Projekte meistens einfache Mechanismen zur Übersetzung der Software. Dadurch lassen sich gute Übersetzungen anfertigen.

[6] um die Software selbst zu benutzen oder um Bekannte zu überzeugen sie zu benutzen, etc.

Bei proprietärer Software hat man das Nachsehen, wenn man zum Beispiel in Ägypten wohnt, und der Hersteller keine arabische Übersetzung anbietet[7]. Es gibt dann keine Möglichkeit eine entsprechend übersetzte Software zu bekommen. Man darf es zum einen nicht selbst machen, weil das gegen die Rechte des Herstellers verstoßen würde, und man kann es zum anderen auch gar nicht selber, weil die Hersteller ihre Übersetzungsmechanismen meistens nicht offen legen.

Wenn bei einer Open Source Software eine Übersetzung in die entsprechende Sprache noch nicht existiert, kann man sich selbst diese Übersetzung erstellen. Durch die einfachen Mechanismen sind dazu oft nicht einmal Programmierkenntnisse oder ähnliches notwendig. Dadurch entstehen Übersetzung für sehr viele verschiedenen Sprachen.

ständige Weiterentwicklung

Kommerzielle Software wird normalerweise all paar Jahre in einer neuen, ausgebauten Version auf den Markt gebracht. Dies ist notwendig, weil sonst die Firmen wegen eingetretener Marksättigung kaum noch Umsatz erzielen würden. Open Source Projekte stehen nicht unter diesem Zwang, da sie mit der Software sowieso keine großen Umsätze erzielen können. Daher werden neuere Versionen mit mehr Funktionen bereit gestellt, sobald diese eingebaut und getestet wurden. Zum Beispiel besaß der Mail–Client Thunderbird im Dezember 2003 noch die Versionsnummer 0.3. Sechs Monate später trug die Software die Versionsnummer 0.7. Durch die ständige Weiterentwicklung kamen die Benutzer nicht erst nach einem oder zwei Jahren in den Genuss der neuen Funktionen, sondern sie bekamen sie, sobald sie verfügbar waren.

Höbere Überlebenschance

Der nächste Vorteil resultiert daraus, dass **diese** nicht kommerzielle Software allen offen steht, somit kann weder ein einzelner Programmierer noch ein Unternehmen die Richtung der Entwicklung vorgeben. Auch Probleme, die bei Anbietern kommerzieller Software entstehen, wenn sie ihre Geschäftstätigkeit einstellen oder von anderen Konzernen übernommen werden, gibt es bei Open Source Software nicht, da ihre Entwicklung und damit ihr Fortbestehen nicht von einzelnen Firmen abhängt. Stellt eine Entwicklergruppe ihre Arbeit ein, kann diese bei genügend großem Interesse von anderen übernommen und weitergeführt werden.

Der Nachwuchs wird gefördert

Ein letzter Vorteil für Programmierer sei noch genannt: es ist für Programmieranfänger gut sich in Programmiercode anderer Programmierer einzulesen und daraus zu lernen. Dies ist bei herkömmlicher Software unmöglich, da ihr Code nur festangestellten (und damit bereits professionell arbeitenden) Programmierern zur Verfügung steht. Somit erleichtert Open Source es zukünftigen Programmierern ihr *Handwerk* zu erlernen und ermöglicht es somit Konzernen „frische“, aber dennoch ausgebildete und programmiererprobte Programmierer einzustellen.

[7] beispielsweise, weil es sich nicht lohnt

5.2 Nachteile

Open Source hat — wie alles andere auch — nicht nur Vorteile. Natürlich gibt es Nachteile und diese müssen selbstverständlich erwähnt werden.

wenig Unterstützung von bekannnten Formaten
: Oft ist es Open Source Programmen, besonders den Office Anwendungen, teilweise unmöglich oder schwierig Dokumente in herkömmlichen Formaten, wie beispielsweise dem Microsoft–Word–Dokument–Format (.doc), zu speichern oder solche zu öffnen. Dies liegt meist daran, dass die Firmen ihre Dateiformate nicht offen legen, weil sie ja mit ihnen Geld verdienen wollen. Diese Probleme werden jedoch nach und nach gelöst. Deshalb hinkt die wohl bekannteste Office Anwendung aus dem Open Source Bereich — nämlich Open Office — meist eine Version hinter dem Microsoft–Word–Dokument–Format hinterher.

teilweise zu komplex
: Open Source Produkte enthalten meist sehr viel mehr Konfigurationsmöglichkeiten als andere Produkte. Dies ist eigentlich ein Vorteil, aber gerade für den Normal–Anwender stellt sich manchmal das Problem, dass er von den Konfigurationsmöglichkeiten „erschlagen“ wird. Da die Standardeinstellungen meistens recht sinnvoll sind, kann man zwar versuchen, nur die Sachen zu konfigurieren, die man versteht. Leider funktioniert diese Methode nicht immer zufriedenstellend, sodass das Programm nur suboptimal — wenn überhaupt — läuft.

keine Einbindung von patentierten Ideen möglich
: Wenn eine Idee patentiert wurde, und der Besitzer das Patent auch nutzt, dann gibt es für Open Source Projekte praktisch keine Möglichkeit diese patentierte Idee in ihr Projekt einzubauen. Eine Lizenz vom Patentbesitzer können sie nicht erwerben, da die Lizenz dann laut den Open Source Lizenzen kostenlos weiter gegeben werden müsste. Also müsste man unendlich viele Lizenzen auf einmal kaufen, damit man diese Idee einbauen darf. Dies ist aber finanziell (fast) niemandem möglich. Hinzu kommt, dass wenn der Patentbesitzer die unendlichen Lizenzen an *ein* Open Source Projekt verschenkt/verkauft hat, er nie mehr Lizenzgebühren von *irgendeinem* Open Source Projekt einfordern kann. Diese können einfach ihre Projekte von dem anderen ableiten, und damit bekommen sie ja auch in den Genuss dieser unendlich vielen Lizenzen.

  Kommerzielle Projekte können aber eine Lizenz pro verkaufte Software erwerben. Das ist zum Beispiel der Grund, warum es keine Open Source Produkte gibt, die GIF–Grafiken erzeugen können. Da Unisys den zur Datenreduktion verwendeten Algorithmus patentiert[8] hat, bleibt die Erzeugung von GIF–Grafiken kommerziellen Programmen vorbehalten, die eine Lizenz dafür erworben haben.

[8] Das erwähnte Patent läuft in Europa und Japan im Sommer 2004 nach 20–jähriger Dauer aus. (Das entsprechende US–Patent ist bereits am 20. Juni 2003 ausgelaufen.) Ob allerdings nicht noch weitere Patente in irgendwelchen Firmenarchiven schlummern, die das GIF–Format betreffen, bleibt abzuwarten.

Verfügbarkeit, beziehungsweise Vorhandensein

Für manches Einsatzgebiet, wie beispielsweise Buchhaltung und Rechnungswesen, Projekt– und Workgroup–Management und verschiedene Bereiche der Branchen–Software gibt es derzeit noch keine ausgereiften Open Source Produkte.

keine / wenig Treiber[9]

Auch ist es oft schwierig Treiber für modernste Hardwarekomponenten für ein Open Source Betriebssystem zu finden, da diese von den Herstellern meist nur für die „normalen“ Systeme, sprich Microsoft Windows Systeme, zur Verfügung gestellt werden.

kein einheitlicher Stil

Als ein letzter Nachteil für Programmierer sei noch folgendes erwähnt: Da der Programm–Code von Open Source Programmen meist von vielen Programmierern geschrieben wurde, ist es teilweise kompliziert sich darin einzuarbeiten. Dies will ich für den Laien noch kurz erklären: Jeder Programmierer hat, ähnlich wie jeder Buchautor, einen eigenen Stil. Jeder Stil hat seine eigenen Vorteile, jeder auch seine Nachteile. Während in Firmen ein „Standardstil“ durchgesetzt wird, welcher aber auch den Programmierern, die einen anderen Stil gewöhnt sind, die Arbeit erschwert, gibt es so etwas bei Open Source Software nicht. Da es aber schwer ist sich in ein Programm einzuarbeiten, in dessen Code verschiedene Stilarten eingearbeitet wurden, ist es oft kompliziert sich in längeren Open Source Codes einzuarbeiten. Dies geht bei sehr großen Projekten, wie zum Beispiel dem Linux–Kernel, soweit, dass sich Teilprojekte abspalten, welche nur den bisherigen Code bearbeiten und dessen Stil vereinheitlichen um den Code so verständlicher zu machen.

[9] Dies betrifft nur Open Source Betriebssysteme (z.B. Linux), nicht aber Open Source Software, die auch Windows Systemen läuft.

5.3 Vorurteile

Nachteilig für Open Source Software wirkt auch, dass viele Benutzer teilweise schlecht bzw. nicht informiert sind. Daraus resultieren einige Vorurteile gegenüber Open Source, die so nicht bzw. teilweise nicht stimmen. Im Folgenden werden diese Vorurteile zuerst einmal aufgezählt und mit einer Erklärung versehen, warum ein großer Teil der Benutzer zu diesen Vorurteilen kommt.

kein Support, keine Garantie
: Da Open Source Software nicht von Firmen entwickelt werden, gibt es keinen Support. Wie einem auch jede Open Source Software beim Installieren oder Starten mitteilt, gibt es „ABSOLUTELY NO WARRANTY“. Man benutzt die Software also komplett auf eigene Gefahr. Die meisten Benutzer zahlen lieber etwas für die Software und haben dann einen Garantieanspruch.

man muss auf Linux umsteigen
: Open Source ist ein Linux/Windows–Wettrennen. Da die meisten aber Windows benutzen, hat Open Source für sie keine bzw. kaum Bedeutung.

Feind des geistigen Eigentums
: Wenn eine Software unter eine Open Source Lizenz gestellt wird, heißt dass, das der Entwickler alle Rechte aufgibt, die ihm laut Urheberrecht eigentlich zustünden.

wird nur von kleinen Firmen verwendet
: Open Source wird nur von kleinen Firmen verwendet. Die großen, wirklich wichtigen Firmen benutzen kein Open Source. Warum sollte dann der Privatanwender Open Source benutzen?

nur für Programmierer nützlich
: Open Source hat nur für Programmierer Bedeutung, da die meisten Benutzer ohnehin niemals den Quellcode ansehen.

man kann damit kein Geld verdienen
: Open Source Software wird umsonst zur Verfügung gestellt. Es gibt keine Möglichkeit damit Geld zu verdienen.

nur eine vorübergehende Bewegung
: Die Open Source Bewegung ist nicht von Dauer. Die Leute werden aufhören, freie Software zu entwickeln, sobald sie sehen, dass andere viel Geld mit ihrer Arbeit verdienen. Und dann werden die Open Source Produkte alle kostenpflichtig.

5.4 Mythen

Die oben genannten Vorurteile — im Folgenden als Mythen bezeichnet — sind teilweise oder komplett falsch, oder es fehlt dem Otto–Normal–Benutzer einfach das Wissen, wie man die Aspekte von Open Source Software im Vergleich zu anderer Software einschätzen soll. Deshalb soll nun hier erklärt werden, warum die Vorurteile so nicht bzw. nicht ganz stimmen.

kein Support, keine Garantie

Es stimmt, dass es keinerlei Garantie gibt, dass das Programm das tut, was es soll. Für eine Fehlfunktion kann der Autor *nicht* verantwortlich gemacht werden. Allerdings ist dies bei kommerzieller Software meistens genauso. Auch die Microsoft Windows XP EULA weißt jegliche Gewährleistung von sich. (siehe auch 4.5.2) Dass man keine Supportansprüche hat, wenn man die Software kostenlos bezieht, versteht sich ja von selbst. Wovon sollen die Entwicklergruppen den Support denn finanzieren? Allerdings gibt es Firmen, die Support–Angebote für Open Source Software zu Verfügung stellen. Bekanntestes Beispiel ist die Firma SUSE, deren Umsatz großen Teils aus dem Support–Angebot besteht.

man muss auf Linux umsteigen

Linux ist zwar auch Open Source, aber man muss Linux nicht benutzen, um Open Source nutzen zu können. Es gibt viele Open Source Produkten für Windows, wie Browser, E–Mail–Clients, Instant Messengers, Textverarbeitungen, Virenscanner, Komprimierprogramme, Compiler und vieles mehr. Man kann sich ohne weiteres ein Windows aufbauen, an dem alles[10] — außer Windows selbst — Open Source ist. Es stimmt allerdings, dass ein Großteil der Open Source Programme für Linux oder andere Open Source Betriebssysteme geschrieben sind.

Feind des geistigen Eigentums

Die Möglichkeit, eigene Entwicklungen unter eine Lizenz zu stellen, die verbietet, dass andere diese Entwicklung verkaufen oder weiterentwickeln, ohne das Ergebnis wieder zu veröffentlichen, ist keine Urheberrechtsverletzung. Vielmehr stellt sie eine aktive Nutzung des Urheberrechtes dar. Wenn ein Entwickler eine Software unter eine Open Source Lizenz stellt, gibt er sein geistiges Eigentum nicht auf, sondern stellt es anderen zu Verfügung. Der ursprüngliche Entwickler (und nur er) kann weiterhin die Software an eine Firma verkaufen, die sie dann kommerziell vermarktet. Oder er kann sie selber vermarkten. Er ist weiterhin der alleinige Copyrighthalter des Originals. Eine Software, die von vielen Entwicklern weiterentwickelt wurde, steht unter dem Copyright von allen Entwicklern und alle zusammen[11] dürfen sie dann auch vermarkten. Schließlich schränkt die Lizenz nicht die Rechte des Entwicklers ein, sondern sie gewährt anderen die Rechte, sie zu benutzen, zu kopieren, zu verändern und vieles mehr.

[10] mal abgesehen von den Treibern für bestimme Hardware

[11] also, wenn jeder einzelne Copyrighthalter einverstanden ist

wird nur von kleinen Firmen verwendet

Open Source wird sehr wohl auch von großen Firmen verwendet. Beispielsweise „beherbergt“ der Open Source Web Server Apache mehr als 60 Prozent der alle Websites weltweit. Darunter befinden sich auch viele der meistbesuchtesten, wie etwa Yahoo. Auch läuft auf einem Großteil der Internetserver sendmail, ein Programm, das die E–Mail–Versendungen koordiniert und ausführt. Fast jede E–Mail kommt einmal an diesem Open Source Programm vorbei. Des weiteren basiert beispielsweise das TCP/IP–Protokoll auf Open Source Code. Das TCP/IP–Protokoll ist das zu Zeit meist genutzte Protokoll, das im Internet und praktisch in jedem Netzwerk benutzt wird.

nur für Programmierer nützlich

Die Vorteile von Open Source hängen sehr stark mit den Vorteilen eines freien Marktes zusammen. Der Wettbewerb zwischen mehreren Anbietern führt dazu, dass man die Software wirklich zum Selbstkostenpreis bekommt und nicht etwas mehr zahlen muss. Er führt zu mehr Innovation und Spezialisierungen, wodurch neue Nischen bei Bedarf gefüllt werden können. Außerdem ist man nicht mehr an einen einzigen Anbieter gebunden. Im Schlimmstfall, kann man seine Probleme selbst lösen, oder, was wahrscheinlicher ist, sich an mehrere Anbieter wenden, die einen besser betreuen können, als das ein einzelner Anbieter jemals können wird.

man kann damit kein Geld verdienen

Dass diese Aussage falsch ist, müsste eigentlich jedem bewusst sein, der schon einmal von Redhat oder SUSE gehört hat. Diese Firmen erzielen einen relativ großen Umsatz dadurch, dass sie zusätzlich zu der Open Source Software auch noch ausführliche Dokumentationen und Support verkaufen. Laut den Lizenzen kann jeder für Support anbieten und dafür Geld verlangen, dieser wird vor allem von Firmen in Anspruch genommen, weil dadurch der eigene Expertenstab verkleinert werden kann. Außerdem können Firmen oder Behörden Aufträge an Entwicklergruppe erteilen, weil sie zum Beispiel in einer Open Source Software noch die eine oder andere Funktion vermissen, oder Spezialllösungen benötigt werden.

nur eine vorübergehende Bewegung

Die heutige Open Source Bewegung hat sich entwickelt, weil andere Programme kostenpflichtig wurden (siehe 2.1 und 2.2). Deshalb wird die Open Source Bewegung kaum vorübergehen, weil die Open Source Entwickler sehen, dass andere viel Geld mit ihrer Arbeit verdienen.

Es könnte allerdings sein, dass der gegenwärtig starke Open Source Trend in Zukunft etwas abflachen wird, aber dass die gesamte Open Source Bewegung vorübergehen wird, ist mehr als unwahrscheinlich.

5.5 Fazit

Die meisten Nachteile beziehen sich nicht auf Open Source Programme im Allgemeinen, sondern auf einzelne Programme, die wegen den Nachteile nicht optimal zu benutzen sind. So stört der Nachteil, dass keine proprietären Dateiformate gelesen werden können, bei beispielsweise einem Mail–Client überhaupt nicht, da alle gängigen Mail–Formate standardisiert und frei verfügbar sind. Der Mail–Client kann also alle gängigen Formate lesen. Ebenso stört die fehlende Möglichkeit, patentierte Ideen einbauen zu können, nur bei Programmen, die auf eine solche Idee angewiesen sind. Die Verfügbarkeit ist meist nur bei Speziallösungen ein Problem.

Der größte Teil der Nachteile bezieht sich dementsprechend nur auf Möglichkeiten, die ein Open Source Programm hat, den Benutzer zufrieden zu stellen. Sobald man aber ein Open Source Programm gefunden hat, das einem alle benötigten Funktionen bereitstellt, entfallen diese Nachteile, sodass das Open Source Programm fast keine Nachteile — dafür aber einige Vorteile — gegenüber einem kommerziellen Programm hat.

6 Open Source als Konkurrenz zu kommerziellen Produkten

6.1 Erfahrungen mit der Konkurrenz Open Source

Nach mehreren Gesprächen mit professionellen Programmierern und Informatikern kamen wir zu folgendem Bild:

Für viele professionelle Programmierer stellt Open Source keine Konkurrenz dar, sondern viel mehr die Möglichkeit zu höheren finanziellen Einnahmen.

Oft werden Libraries aus Open Source Projekten, die unter der GNU Lesser GPL stehen, mit in Projekte eingebunden, was viel Programmieraufwand und damit finanzielle Mittel spart. Zum Ausgleich werden eigene Libraries, wenn auch nicht die ganzen Programme (man will schließlich an deren Lizenzen verdienen), wiederum unter die GNU Lesser GPL gestellt und für Open Source Projekte zur Verfügung gestellt. Selbiges trifft auf Programme zu, deren Entwicklung aus finanziellen oder personellen Problemen abgebrochen wird. Sie werden auch unter die Lesser GPL gestellt und veröffentlicht. Auch werden oft Gesamtleistungen angeboten, bei deren Durchführung mehrfach auf Open Source Programme oder Ideen zurückgegriffen wird. Beispielsweise werden bei der Datenverwertung oft SQL Datenbanken, die Script Sprache Perl oder andere Datenverarbeitungshilfsmittel benutzt. Vor allem kleine Computer–Wartungs–Firmen verdienen mit dem Einrichten und Warten von Open Source Systemen ihr Geld. Ein Paradebeispiel hier sind Linux Router oder Automatische–Daten–Backup–Server, sowie den Support für Open Source Programme.
Auch sind viele professionelle Programmierer in ihrer Freizeit damit beschäftigt Open Source Lösungen für andere Probleme zu finden. „Es ist sowohl Hobby als auch Beruf“. Manch ein Programmierer hat auch nur deswegen seinen Job bekommen, da er einen gewissen Ruf in „der Open Source Szene“ und einige von ihm geleitete Projekte vorweisen konnte.

Leider gibt es auch andere Erfahrungen mit Open Source Anwendungen. Besonders kleinere Firmen, die rein auf das Programmieren kleiner Programme, Tools und Scripte spezialisiert sind, haben große Probleme die Konkurrenz durch Open Source Lösungen zu überstehen. Doch auch große Firmen sprechen von „großen Gewinneinbrüchen“ auf Grund von neuen Open Source Alternativen, die den „Preis zerstören“, besonders Microsoft klagt mittlerweile über Probleme durch Linux & CO.

6.1.1 Zusammenfassung

Auch wenn viele Firmen und Open Source Gegner gerne von Open Source als Arbeitsplatzvernichter sprechen, kamen wir eher zu dem Bild, dass Open Source zwar manchen Arbeitsplatz bedroht oder gar zerstört, aber an anderen Stellen gerade für kleine Unternehmen neue schafft.

6.2 Open Source als Alternative

Kategorie	Proprietäres Programm	Open Source Programm
Betriebssysteme	Windows	Linux
Internet Browser	Microsoft Windows Internet Explorer	Mozilla & Firefox
Mail–Clients	Microsoft Outlook	Mozilla & Thunderbird
Komprimier–Programme	WinAce, WinZip, WinRar	7–Zip, Uharc
C++–Compiler	Microsoft Visual C++	Bloodshed Dev–C++, gcc[12]
Firewalls	Kerio Personal Firewall, Microsoft Windows XP Firewall	SmoothWall Express 2.0
Offline Browser	Offline Explorer, JOC Web-Spider	HTTrack Website Copier
Office–Programme	Microsoft Office	OpenOffice
Instant Messenger	ICQ, Trillian	Miranda Instant Messenger
Virenscanner	Norton Virusscanner, AntiVir	DansGuardian AV Scanner, OpenAntivirus, ClamAV
Programme zum Erstellen von PDFs	Adobe Acrobat	GhostView + GhostScript, LaTeX
Videobearbeitung	Ulead Video Studio	VirtualDub[13]
Musikbearbeitung	Magix Music Studio deLuxe	Audacity
Brennprogramme	Nero Burning Room, Record-Now	Cdrdao
Video– u. Musikwiedergabe	Windows Media Player, Winamp	VLC

[12] **G**nu **C**ompiler **C**ollection; eine Sammlung freier Compiler
[13] Nur zum Schneiden geeignet; Nicht zum Vertonen, Titeleinblendung etc.

6.2.1 Vergleich: Mozilla Firefox u. Microsoft Internet Explorer

Im Folgenden soll ein Open Source Internet Browser mit einem proprietären Internet Browser verglichen werden. Es handelt sich demnach um eine Programmart, die von einem großem Teil der Bevölkerung täglich benutzt wird. Als Open Source Kandidat wurde Mozilla Firefox gewählt, weil er ein relativ weit verbreiteter Open Source Browser ist. Für die proprietäre Lösung wurde der Microsoft Internet Explorer gewählt, weil er bei Windows standardmäßig installiert ist, und deshalb von dem größten Teil der Internet–Surfer benutzt wird.

Mozilla Firefox **Vorteile**	**Microsoft Internet Explorer** **Nachteile**
Der Open Source Browser hat bzw. hatte weniger bekannte Sicherheitslücken. Keine der bekannten Sicherheitslücken, die im c't–Browsercheck[10] veröffentlicht wurden, funktioniert mit der aktuellen Version. Dies mag teilweise daran liegen, dass dieser Browser für Angreifer, die gezielt die Sicherheitslücken ausnutzen, weniger attraktiv ist, weil er nicht so oft benutzt wird, wie beispielsweise der Microsoft Internet Explorer. Auf jeden Fall ist aber festzustellen, dass sobald eine der wenigen Sicherheitslücken bekannt geworden ist, in den folgenden Tagen eine Version zum Download bereit stand, die nicht mehr anfällig war.	Der Microsoft Internet Explorer 6 auf einem Windows XP–System, auf dem *alle* verfügbaren Sicherheitsupdates von Microsoft eingespielt sind, besitzt zur Zeit[14] immer noch bekannte Sicherheitslücken, mit denen sich kompletter Zugriff auf den Computer erlangen lässt, dazu muss man lediglich eine entsprechend präparierte HTML–Seite laden. Auf heise.de steht ein Browsercheck[10] zur Verfügung mit dem man testen kann, wie viele Sicherheitslücken der eigene Browser noch besitzt. Meistens ist die einzige Möglichkeit Active–X auszuschalten, womit allerdings der größte Vorteil des Internet Explorers gegen über Mozilla Firefox verloren geht.
Da Mozilla Firefox ein Open Source Programme ist, und zusätzlich über ein ausgeklügeltes Erweiterungssystem verfügt, lässt sich der Browser an jede individuellen Ansprüche anpassen. Es gibt zum Beispiel eine Erweiterung, womit sich der Browser komplett über Mausgesten steuern lässt, oder eine die anzeigt, wie lange man schon surft, oder, oder, oder ...	Erweiterungen lassen sich zwar über Active–X installieren, aber da jedes Active–X–Controll vollen Zugriff auf den gesamten Internet Explorer hat, kann ein Fehler in einer Erweiterung durchaus dazu führen, dass das Gesamtpaket weitere nennenswerte Sicherheitslücken erhält.

[14] Stand: 11. Juni 2004

Mozilla Firefox **Vorteile** Fortsetzung...	**Microsoft Internet Explorer** **Nachteile** Fortsetzung...
Mozilla Firefox besitzt einige nützliche Features, wie zum Beispiel einen Download–Manager, einen Popup–Blocker (der die lästigen Werbe–Popups verschwinden lässt), oder Tab–Browsing (was ermöglicht, neue Internetseite im Hintergrund zu öffnen). Außerdem unterstützt er alle gebräuchlichen Internet–Standards, wie zum Beispiel Umlaute in Internet–Adresse. Firefox ist ungefähr halb so groß wie der Internet Explorer, und arbeitet daher auch deutlich ressourcensparender. Daraus resultiert auch, dass die durchschnittliche Ladezeit pro Seite unter der des Internet Explorers liegt.	Der Internet Explorer beherrscht diese Features nicht. Am schlimmsten ist eigentlich, dass er den seit April 2003 veröffentlichten Standard RFC 3490[21] immer noch[15] nicht unterstützt. Das bedeutet, dass es nicht möglich ist, auf eine Internetseite mit Umlauten zuzugreifen (Bsp.: `http://www.baden-württemberg.de`, `http://www.tübingen.de`, `http://www.österreich.de`). Meistens führt es zwar auch zum Ziel, wenn man ä mit ae usw. ersetzt, aber das ist kein allgemeines Rezept, da das nur funktioniert, wenn beidee Namen auf die gleiche Seite verweisen, was nicht zwangsläufig der Fall ist. Microsoft hüllt sich bisher in Schwiegen, ob und wann ein Update erwartet werden darf. Es gibt zur Zeit nur eine Möglichkeit, den Internet Explorer zu „überreden“; nämlich ein Active–X–Plugin. Dieses wurde von der IDN Open Source Software Group[11] entwickelt und ist erst in der Alpha–3–Testversion verfügbar[16].

[15] Stand: 15. Juni 2004

[16] Wie der Name den Entwicklergruppe bereits andeutet, handelt es sich hierbei um ein Open Source Projekt

Mozilla Firefox **Nachteile**	**Microsoft Internet Explorer** **Vorteile**
Der Browser lässt sich nicht für Windows–Updates oder sonstige Updates von Microsoft verwenden. Allerdings liegt das nicht am Browser, sondern daran, dass Microsoft unbedingt den Internet Explorer benutzen lassen will.	Er ist der einzige Browser, mit dem man ein Windows–Betriebssystem updaten kann. Deshalb kann niemand auf ihn verzichten.
Mozilla Firefox ist nicht der weitverbreiteste Browser. Deshalb achten manchmal die Homepage–Ersteller kleinerer Seiten nicht darauf, dass die Seiten sich mit allen Browsern betrachten lassen. Dieses Problem gibt bzw. gab es zum Beispiel auf der Seite `http://www.smvkepi.de`. Am 06.03.2004 wurde im Gästebuch auf dieses Problem hingewiesen, und mittlerweile[17] wurde es nur notdürftig gelöst.	Der Internet Explorer ist der weitverbreiteste Browser. Das bedeutet, dass eine Homepage sich mit ihm eigentlich immer so anzeigen lässt, wie der Ersteller sich das vorstellt. Allerdings unterstützt er manche Features wie zum Beispiel die Verschachtelung von `<div></div>`–Anweisungen nicht. Dies führt dazu, dass wenn sich ein Fehler im Homepage–Design befindet, der jeweilige Ersteller das nicht merkt und die Seite sich deshalb nur im Internet Explorer „korrekt“[18] anzeigen lässt.
Es wird kein Active–X unterstützt. Allerdings muss man sich ernsthaft überlegen, ob man Active–X überhaupt braucht. Ohne Active–X funktionieren ein großer Teil der Dialer nicht mehr. Manche Online–Spiele funktionieren leider auch nicht mehr. Leider sind auch ein Teil der gut–gemeinten Programme, wie zum Beispiel manche Online–Viren–Scanner, betroffen. Für alle anderen Zusatzfunktionen besitzt Firefox ein eigenes Erweiterungssystem, sodass hier Active–X überhaupt nicht benötigt wird.	Der Microsoft Internet Explorer unterstützt Active–X. Dadurch lassen sich beliebige Zusatzfunktionen im Browser realisieren. Wer auf Seiten surft, die unbedingt ein Active–X–Controll brauchen (wie zum Beispiel `http://www.windowsupdate.com`, oder viele Seiten mit kostenpflichtiger Zugangssoftware[19]), kommt um den Microsoft Internet Explorer nicht herum.

[17] Stand: 13. Juni 2004

[18] „korrekt“ bedeutet hier, so wie der Ersteller sich das Ergebnis vorstellt und nicht so, wie es rein logisch angezeigt werden müsste.

[19] s.g. Dialer

Alles zusammen betrachtet, stellt Mozilla Firefox eine durchaus ernstzunehmende Alternative zum Internet Explorer dar. Auch für einen Benutzer, den das Hauptproblem des Internet Explorers — die viele verschiedenen Sicherheitslücken — nicht soviel Kopfschmerzen bereitet, wie einem großen Teil der Programmierern, der Open Source Community, eigentlich aller, die sich intensiver mit Computer und dem Internet beschäftigen als der Otto–Normal–Benutzer.

Dass der Internet Explorer trotzdem eindeutig Marktführer ist, kann eigentlich nur dadurch erklärt werden, dass er bei Windows standardmäßig installiert ist, und viele zu faul sind, einen zweiten Browser zu installieren. Evtl. könnte es auch daran liegen, dass ein Großteil sich überhaupt nicht bewusst ist, wie unsicher der Internet Explorer eigentlich ist. Auch scheint die Ansicht verbreitet zu sein, der Internet Explorer könne — durch seinen großen Marktanteil — überhaupt nicht so schlecht sein. Dagegen lässt sich ein nicht ganz ernstgemeintes Beispiel aus der Biologie anführen: „Millionen von Fliegen ernähren sich von Kot. So viele können sich doch nicht irren?“

6.2.2 Fazit

Für eigentlich alle Bereiche des täglichen Programmbedarfs gibt es Alternativen aus dem Open Source Bereich. Es hapert manchmal an der Spezialsoftware, die der Otto–Normal–Benutzer nicht benötigt, sondern nur von einer kleinen Gruppe gebraucht wird. So kann man beispielsweise nach einem Open Source Programm zur kompletten Videobearbeitung vergeblich suchen. Oder bei der Buchhaltung, dem Rechnungswesen, der Verwaltung von Baugenehmigungen, Friedhöfen, Schulen, etc. kann man nicht oder nur beschränkt auf Open Source Alternativen zurückgreifen.

Es besteht allerdings die Hoffnung, dass durch die immer größere Verbreitung von Open Source, diese Nischen bald geschlossen werden können. Hier kommt ja genau ein Vorteil von Open Source zum Tragen: Sobald jemand sich ein solches Programm erstellt hat, können es alle benutzen[20].

[20] Vorausgesetzt es wird als Open Source veröffentlicht.

7 Der behördliche Einsatz von Open Source

7.1 Israel suspendiert Microsoft–Verträge

Wie *The Register[19]* am 24.11.2003 berichtete, hat das Israelische Finanzministerium alle Verträge mit Microsoft auf Eis gelegt. Dies geschah als Reaktion darauf, dass Microsoft vom israelischen Kartellamt offiziell als Monopolist eingestuft wurde. Die Suspendierung soll auf jeden Fall das ganz Jahr 2004 andauern. Dies bedeutet, dass keine israelische Behörde neue Software oder Software–Updates wie beispielsweise auf MS Office 2003 von Microsoft kaufen darf. Als Ersatz soll OpenOffice verwendet werden. Das Arbeitsamt vollzieht den ersten Schritt und ersetzt an über 75% seiner Arbeitsplätze das Microsoft–Produkt durch OpenOffice. Als Betriebssystem wird zunächst weiterhin das schon vorhanden Windows benützt.

Microsoft gab dazu einen seiner üblichen Kommentare:

> „Das Arbeitsamt hat ein unreifes und noch nicht bewährtes Software–Paket ausgewählt und dessen Funktionalität ist bestenfalls auf dem Stand von Office 97."

Diese Funktionalität scheint dem Arbeitsamt aber zu genügen, da ja die Möglichkeit besteht, *beliebige* Anpassungen in Auftrag geben zu können. Der Ausgangspunkt scheint die Tatsache zu sein, dass verschiedenen Microsoft–Applikationen für Macintosh die Unterstützung von Hebräisch und anderen von rechts nach links geschriebenen Sprachen vermissen ließen. Allerdings war Microsoft offenbar nicht willens, die Applikationen anzupassen. Mit OpenOffice ist dies natürlich kein Problem mehr, da jeder sich die Unterstützung selber einbauen bzw. einbauen lassen kann. So unterstützen viele Open Source Produkte mehr Sprachen als kommerzielle Produkte, da lokale Entwicklergruppen die entsprechenden Übersetzungen entwickeln.

7.2 Linux in München

Die Stadt München[14] hat am 28.05.2003 beschlossen, künftig sowohl beim Betriebssystem für ihre rund 14.000 Computer wie auch bei der Office–Software auf Open Source Produkte setzen. Der Stadtrat verabschiedete diese Entscheidung mit der absoluten Mehrheit von SPD und GRÜNEN, denen sich alle Fraktionen mit Ausnahme der CSU anschlossen.

Die Stadt hatte ein Gutachten in Auftrag gegeben, welches die Vor– und Nachteile einer Umstellung vom derzeitigen Standard „Windows NT/Microsoft Office" alternativ auf „Windows XP/Microsoft Office" oder „Linux/OpenOffice" bewerten sollte. Dabei ergab das Gutachten einen klaren strategisch–qualitativen Vorsprung für die Open Source Lösung.

In einem weiteren Schritt sollen nun die Umsetzungsmöglichkeiten in einem Feinkonzept ermittelt werden. Auf dessen Basis hat der Stadtrat dann beschlossen, wie die Umstellung auf Linux/OpenOffice genau vollzogen werden soll.

Interessant war vor allem die Vorgeschichte bis es zu der Entscheidung 28.05.2003 kam.

7.2.1 Vorgeschichte

03.04.03: Der Stadtrat berät die im April 2002 in Auftrag gegebene Studie. Die Studie empfiehlt die Einführung von Linux auf den 16.000 Desktops und Servern die Stadtverwaltung.

04.04.03: Angeblich[9][1][18] unterbrach Microsoft–CEO Steve Ballmer sogar seinen Winterurlaub, um mit Münchens Oberbürgermeister Christian Ude (SPD) über Preisrabatte zu verhandeln. Unter anderem scheint Microsoft bereit zu sein, sich im Rahmen des Programms „Schulen ans Netz“ zu engagieren – und damit die Stadt finanziell erheblich zu entlasten.

Offenbar befürchtet Microsoft einen Domino–Effekt, wenn Linux in München eingeführt wird, und ist deshalb bereit, bei den Lizenzkosten Zugeständnisse zu machen, um die Kommunen auf seiner Plattform zu halten.

Die Entscheidung wird auf den 28. Mai 2003 vertagt.

10.04.03: Jürgen Gallmann, Chef von Microsoft Deutschland, schloss nach langen persönlichen Gesprächen mit Regierungsvertretern in Berlin neue Rahmenlizenzverträge mit Bundesinnenminister Otto Schily. Sie sollen der gesamten öffentlichen Verwaltung künftig die Beschaffung von Microsoft–Produkten zu Sonderkonditionen ermöglichen. „Durch diese Vereinbarungen sparen Bund, Ländern und Gemeinden viel Geld“, hofft der SPD–Politiker.[9]

BundesTux[1] sah Microsoft nach der Linux–empfehlenden Studie in München unter Zugzwang und sieht in der Reaktion eine Bestätigung für Linux und Open Source:

> „Wenn Linux nicht wirklich eine Alternative wäre, würde sich Microsoft wohl kaum zu solchen Schritten genötigt sehen. [...] Jedem logisch denkenden Menschen ist klar, dass sich die Haushalte der Kommunen am besten durch gar keine Lizenzgebühren entlasten lassen.“

16.04.03: Die ÖDP–Stadträtin Mechthild v. Walter stellt den Antrag „aus Kosten– und Sicherheitsgründen von Windows NT auf Linux“ zu wechseln. [23]

Die Stadt München muss – trotz erfolgreicher Verwendung von Windows NT – auf ein neues Betriebssystem umstellen, weil die Herstellerfirma Microsoft Windows NT nicht mehr unterstützt. Wenn jetzt wieder ein Microsoft–Betriebssystem verwendet wird, steht die Stadt in ein paar Jahren vor dem gleichen Problem. Deshalb zahlt sich auch ein von Microsoft nochmals reduzierter Einstandspreis mittel– und langfristig nicht aus.

Noch entscheidender ist für die ÖDP–Stadträtin die Daten– und Benutzersicherheit:

> „Windows XP übermittelt automatisch Benutzerdaten über das Internet an Microsoft und kann nur schwer auf 'versteckte' Funktionen überprüft werden. Diese Intransparenz dürfte auch Geheimdiensten die allgegenwärtige Industriespionage erleichtern.“

22.04.03: Gegenüber Linux–Muenchen.de[12] bestätigt SUSE Verhandlungen mit der Stadt München. Allerdings will SUSE sich nicht weiter dazu äußern.

19.05.03: Linux–Muenchen.de[12] stößt auf die von der Unilog Integrata angefertigten „Ergänzungen zum Abschlussbericht".

Diese Ergänzung plädiert nun für eine reine Microsoft Lösung. Auf 29 Seiten wird die zur ersten Studie völlig konträre Empfehlung wie folgt begründet:

- Gegen eine einmalige Zahlung von Euro 250.000 sei Microsoft zudem bereit, die Produktunterstützung für Windows XP bis Dezember 2009 zu verlängern. Innerhalb von fünf Jahren ab der Entscheidung ist deshalb von Kosten einer Folgemigration nicht auszugehen.

  (Evtl. nötige Folgemigrationen nach dieser Zeit werden in der Studie nicht behandelt.)

- Durch den Einsatz des neuen Microsoft Produktes „Virtual PC" könne aus einer „harten Migration" eine „weiche Migration" werden. Die Kosten für „Virtual PC"–Lizenzen werden zwischen Euro 300.000 und Euro 810.000 beziffert, was laut der Studie im Gesamten zu einer Kostensenkung durch Emulation alter Umgebungen führen würde.

  (Produktbeschreibung von Microsoft: „Mit Microsoft Virtual PC 2004 können Anwender verschiedene Betriebssysteme komfortabel auf einem einzigen Computer betreiben. Dabei lässt sich zwischen Betriebssystemen mit einem Mausklick so einfach hin– und zurückschalten wie sonst nur zwischen Anwendungen. So lassen sich komplexe Multiboot–Konfigurationen vermeiden, wenn Anwender während der Arbeit mehrere Betriebssysteme benötigen. Virtual PC bildet einen Computer so genau nach, dass installierte Anwendungen die virtuelle Maschine, in der sie betrieben werden, nicht von einem echten Computer unterscheiden können. Änderungen an der virtuellen Maschine haben keine Auswirkungen auf den tatsächlichen Computer.")

- Durch Preisnachlässe seitens Microsoft wird die Microsoft–Lösung gegenüber der Studie vom 23.01.2003 um ca. 15% billiger. (Euro 34,18 Mio → Euro 29,3 Mio)

21.05.03: Financial Times Deutschland[5] berichtet, dass „Der US–Softwarekonzern Microsoft [...] kurz davor [steht], den prestigeträchtigen Auftrag zur Ausrüstung der Münchner Stadtverwaltung zu gewinnen“, und beruft sich dabei auf Aussage der SPD im Stadtrat.

Außerdem fühlen sich viele Linux–Fans betrogen. So schreibt zum Beispiel *kai* auf MacQuardians[13]:

> **Die ursprünglich so vielversprechende europaweite Initiative „Linux in die Ämter“ stellt sich nach und nach (zumindest in Deutschland) als Farce heraus. Neuester Schwank in der Seifenoper mit dem Pinguin als dummem August in der Hauptrolle: München. Es ist natürlich so gekommen wie ich befürchtet habe: München droht mit Linux–Umstieg und macht ein bisschen Rambazamba, M$-Supersalesman Steve „Schwanensee“ Ballmer kommt vorbei und verhandelt und schwupps, schon werden die großen Rabatte gewährt und Windows ist auf einmal die billigere Lösung!**
>
> Vergessen sind die Reden vom alles abtötenden Monopolisten, den man nicht unterstützen darf und von der Abhängigkeit, in die man sich begibt; nichts weiß man mehr von der Gleichberechtigung, der Offenheit und der Wahlfreiheit, die nur Open Source ermöglicht! Vorbei ist's mit dem Wirtschaftsstandort Deutschland/Europa, den man per lokal entwickelten und gemeinsam genutzten Open Source Programmen kollektiv fördern und so das Geld im Lande halten kann um der amerikanischen Dominanz einen Gegenpol zu liefern! Sicherheit und Immunität gegen das riesige Spektrum der M$-basierten Hacks und Exploits scheint auch nix mehr zu zählen! „Längerfristig denken“, was heißt das? Keine Ahnung, fragen wir doch mal Münchens OB Christian Ude und Kollegen (Schreibt, Leute, schreibt! Am 28.Mai wird die finale Entscheidung getroffen, vielleicht können wir ja noch was bewirken!)
>
> ...

Zum Verständnis:

Pinguin: Tux, das Maskottchen von Linux, ist ein Pinguin

M$: Verbreitete Abkürzung für MicroSoft, die auf Geldgier (Dollar–Gier) anspielt

Steve „Schwanensee“ Ballmer: Steve Ballmer unterbrach seine Urlaub, um mit München zu verhandeln. (siehe 04.04.03)

„Längerfristig denken“: zielt auf den Antrag der ÖDP–Stadträtin Mechthild v. Walter ab (siehe 16.04.03), in dem erklärt wird, dass Microsoft–Produkte mittel– und langfristig nicht auszahlen, weil man in ein paar Jahren durch das Einstellen des Supports von Microsoft vor dem gleichen Problem stehen würde.

23.05.03: Das Migrationsprojekt wird im *nicht*–öffentlichen Teil besprochen. Entscheidung am 28.05.03 wird ebenfalls nicht öffentlich sein.
(Die (teilweise) Veröffentlichung der Studien fand erst nach der Entscheidung am 28.05.03 statt. Die waren zu diesem Zeitpunkt der Öffentlichkeit also nur sehr beschränkt zugänglich.)

25.05.03: Die Fraktion der SPD beschließt, am 28.05.03 für Linux/Open Source zu stimmen. So heißt es in der Presseerklärung[24]:

> „Wir sind uns voll darüber bewusst, dass unsere Entscheidung Signalwirkung hat. Deshalb haben wir uns intensiv mit der Materie auseinandergesetzt. [...] Schließlich führte die erneute Gesamtbewertung [...] zum Gleichstand zwischen beiden Lösungsvarianten. Da aber die Kombination Linux und Open–Source–Office–Lösung qualitativ–strategisch eindeutig die Nase vorn hat, entschied sich die SPD–Fraktion für diese Alternative als langfristige Weichenstellung."

26.05.2003: Die Grünen werden auch für die Linux/Open Source–Lösung stimmen. Neben den bereits benannten Gründen steht in der Presseerklärung[27] ein weiterer Punkt:

> „Auch der IT–Standort München wird mit dem Beschluss gestärkt werden, denn die Umstellung auf Linux wird städtische Aufträge nach sich ziehen und so qualifizierte Arbeitsplätze schaffen."

28.05.2003: Der münchner Stadtrat beschließt künftig sowohl beim Betriebssystem für die rund 14.000 Computer wie auch bei der Office–Software auf Open Source Produkte setzen. Der Stadtrat verabschiedet diese Entscheidung mit der absoluten Mehrheit von SPD und GRÜNEN, denen sich alle Fraktionen mit Ausnahme der CSU anschließen.

In den folgenden Tagen: Damit könnte die Entscheidung eigentlich gefallen sein, aber Microsoft erwägt ernsthaft[17][9], mit Hilfe der Münchner CSU gegen die Stadt zu klagen, weil es keinen fairen Wettbewerb gegeben hätte. Nachdem die Münchner SPD in ihren Presseerklärung[25] unter dem Titel „Linux–Entscheidung: Rathaus–SPD gibt CSU Nachhilfe" erklärt, dass der „Grundsatzbeschluss des Stadtrats [...] rein gar nichts mit der Auftragsvergabe an eine Firma zu tun [hat]." Die Firma Microsoft könne sich genau so wie jede andere Firma bei der Vergabe des Auftrags für die Linux–Umstellung bewerben. Danach rudern sowohl die CSU wie auch Microsoft zurück[17][9]. Angeblich sei dem zuständigen Redakteur von Capital bei dem Artikel eine „Panne" unterlaufen ist.

7.2.2 Entwicklung

02.12.2003: Die Münchner SPD erklärt in ihrer Presseerklärung[26], dass „viele anfangs aufgeworfene Probleme [...] bereits gelöst [sind.] Die Erarbeitung des Feinkonzepts für die Migration läuft gut." Zusätzlich zeigen mehrere Seminare, dass sich „vor allem kleine und mittlere Unternehmen [...] eingehend mit der Entwicklung in der Landeshauptstadt [beschäftigen] und erwägen die Migration auf Linux."

09.01.2004: Die Computerwoche Online[2] berichtet, dass die Landeshauptstadt mit ersten Problemen kämpft. So stellt zur Zeit vor allem die Tatsache, dass mehrere kleine Spezialprogrammen (Software zum Verwalten von Baugehmigungen, Friedhof etc...) wenig bis gar für Linux verfügbar sind. Allerdings ist man der Meinung, dass diese Probleme lösbar sind.

25.05.2004: Die Münchner GRÜNEN berichten in ihrer Presseerklärung[28], dass das Feinkonzept fertig erstellt ist: „Linux für München wird funktionieren. [...] Die Umstellung auf Linux und Open Source Anwendungen soll bis Ende 2008 abgeschlossen sein." Wichtig sei nun, dass man nicht vom einem Monopolisten (Microsoft) zum nächsten (IBM oder SUSE) gehe, sondern dass die „viele kleine und nicht so kleine Fachanwendungen und Speziallösungen" von „kleinen und mittelständischen IT–Firmen, die in München und der Umgebung ansässig sind," erstellt werden, und so der IT–Standort München gestärkt wird.

16.06.2004: Der Stadtrat beschließt[9], dass das Projekt LiMux[21] starten kann. Der Stadtrat hat den Plan, der seit der Entscheidung am 28.05.2003 erarbeitet worden ist, und der die Umstellung der gesamten Computerlandschaft beschreibt, nun offiziell verabschiedet. Wie bereits bei der Vorentscheidung stimmten bis auf die CSU alle Parteien dafür. Im Einzelnen soll die Migration in drei Schritten erfolgen: Zunächst sollen alle noch mit Windows NT–laufenden Rechner mit Open Office und Mozilla[22] ausgerüstet werden. „Vorab steht die Umwandlung der rund 7000 Office–Makros für Formulare wie Urlaubsanträge oder Reisekostenabrechnungen an, die damit endlich zentralisiert werden können", freut sich Jens Mühlhaus von den GRÜNEN. 2005 und 2006 geht es dann an die Migration zum neuen Betriebssystem Linux, das letztendlich komplett mit freier Software arbeiten soll. Bis 2008 steht dann die schwierige Anpassung der Fachanwendungen an, für die laut Mühlhaus Kreativität und eine gute Zusammenarbeit zwischen der Verwaltung und Open Source Entwicklern erforderlich ist. Für Mühlhaus gibt es aber noch einen kleinen Wermutstropfen: Die Münchner Schulen sollen erst in zwei Jahren auf Linux umgerüstet werden, sodass die Auszubildenden bis dahin noch mit der „Windows–Welt" aufwachsen. Microsoft biete für den Bildungssektor „sehr billige Lizenzen an". Da falle es schwer, den politischen Willen zur raschen Migration zu bündeln.

[21] LiMux: offizieller Name des Migrations Projektes: **linux** + **M**ünchen
[22] d.h. mit einer Open Source Textverarbeitung und einem Open Source Browser mit E–Mail–Client

7.3 Fazit

Immer mehr Kommunen und Behörden schielen in Richtung Open Source. Das liegt vor allem daran, dass von kommerzieller Software für jeden Rechner eine Lizenz gekauft werden müsste. (Die Stadt München hätte 14.000 mal Windows XP, Office und jede weitere Software kaufen müssen.) Eine Open Source Lösung muss zwar einmal zusammen gestellt werden, darf dann allerdings auf beliebig vielen Rechnern installiert werden.

München wurde von uns als Beispiel herausgegriffen, aber auch viele andere machen sich Gedanken über eine Migration auf Open Source Software. Die meisten sind noch nicht so weit, und wollen anscheinend auch nicht gerade unter den Ersten sein, die die Migration durchführen. Zum Beispiel findet man bei Linux–Muenchen.de[12] folgende Städte/Behörden die Ambitionen in Richtung Open Source zeigen bzw. bereits umstellen/umgestellt haben (alphabetisch sortiert):

- Auswärtiges Amt
  (Linux wird wegen Kosten und Sicherheitsaspekte benutzt)
- Bayern
  (SPD wirbt im Wahlkampf mit "Mehr Linux – mehr Freiheit")
- Bayrische Vermessungsämter
  (CSU gibt vollendete Umstellung auf Linux von ca. 3000 PCs bekannt)
- Europäisches Parlament
  (Eine Gruppe europäischer Politiker wollen aus Kosten– und Sicherheitsgründen auf Open Source Software umsteigen.)
- Ingolstadt
  (ÖDP macht sich für Linux stark)
- Mannheim
  (Stadtverwaltung berät über mögliche Umstellung)
- Monopolkommission
  (Auf den Clientsystem wurde eine Debian–Linux–Version installiert)
- München
  (s.o.)
- Neu Delhi
  (Opposition für Open Source; Open Source als Alternative)
- New York
  (Stadtverwaltung berät über Open Source Software)
- Polizei Niedersachsen
  (Im September wurde damit begonnen, 11.620 Arbeitsplätze mit Linux–PCs auszustatten.)
- Rheinland–Pfalz: Neun Kommunen
  (Die Stadtverwaltungen aus Alzey, Kaiserslautern, Koblenz, Landau, Mainz, Neustadt/Weinstraße, Speyer, Trier und Worms erwägen gemeinsam auf Open Source Produkte umzusteigen.)
- Stuttgart
  (Grüne setzen sich für Linux ein)
- Wien
  (350 Server umgestellt; einzelne Stimmen aus der SPÖ–Regierung befürworten Linux auf den Clients)
- Zürich
  (Grüne und SP stellen Antrag für Open Source)

8 Szenendarstellung

Oft wird von *Außenstehenden* von einer „Open Source Szene“ geredet, und diese einfach als die „Computer Szene“ begriffen. Um die „Szene“ gibt es viele Gerüchte und die meisten sehen in ihr wohl eine Sammlung von illegalen Computerfreaks und Linux–Begeisternden, die an kommunistische oder anarchistische Überzeugungen glauben. Im Folgenden soll ganz kurz erläutert werden, wovon wir reden, beziehungsweise wovon wir **nicht** reden, wenn wir von einer „Open Source Szene“ reden. Als erstes soll festgestellt werden, dass wenn man von einer „Szene“ im Computerbereich redet, **keine** feste Gruppierung mit festen Regeln oder ähnlichem gemeint ist, sondern lediglich eine Art verstreute „Sammlung“ von Menschen, die in bestimmten Bereichen mehr oder weniger den gleichen Zielen folgen. Ist von Mitgliedern dieser „Szenen“ die Rede, sind Personen, die sich eben diesen Zielen zuordnen, gemeint. Es sollen nun ein paar „Szenen“ dargestellt werden:

8.1 Die „Hacker und Cracker Szene“

Unter der „Hacker und Cracker Szene“ verstehen viele, die mehr oder weniger zusammengehörigen Hacker und Cracker, die das Ziel haben, entweder soviele Informationen, wie möglich, einer möglichst großen Öffentlichkeit zugänglich zu machen, oder sich selbst zu bereichern, illegale Kopien und Cracks von und für alle Programme und Spiele zu ermöglichen oder gar einen anarchistischen Zustand im Netz zu erreichen. In wie weit man an eine solche „Szene“ glauben mag, soll genauso jedem überlassen sein, wie die Frage, was eine solche „Gruppierung“ wirklich erreichen will, doch eins soll klar gesagt sein: sollte diese „Hacker und Cracker Szene“ existieren, ist sie **nicht** die „Open Source Szene“, auch wenn viele Konzerne das in ihrer Propaganda anders darstellen. Es stimmt, dass die „Open Source Szene“ viel mit einer „Hacker und Cracker Szene“, falls diese in einer „festen“ Form existiert, gemein haben würde. Beispielsweise ist es unter Hackern meist üblich geschriebenen Code frei zur Verfügung zu stellen und viele Hacker und Cracker benutzen Open Source Programme zu ihrem eigenen Vorteil. Aber es bleibt dabei, dass sich „Mitglieder“ der „Open Source Szene“ sich explizit von einer „Hacker und Cracker Szene“ distanzieren.

8.2 Die „Ripper Szene“

Der Begriff einer organisierten „Ripper Szene“ fiel erstmals während eines Interviews über Kopien von Audio CDs und Video DVDs mit einem Film–Industrie Vertreter. Dieser versuchte vor der Gefahr so einer großen fest strukturierten Gruppierung zu warnen, welche das Ziel habe der Musik– beziehungsweise Film–Industrie so viel wie möglich zu schaden. Abgesehen davon, dass meines Wissens nach eine solche Gruppierung nicht existiert, hätte auch eine solche Gruppierungen nichts mit der „Open Source Szene“ zu tun, außer dass auch Ripper, also Personen die DVDs kopieren und komprimiert verteilen, gerne auf Open Source Programme zurückgreifen und meist selbst den Code ihrer Programme frei zur Verfügung stellen. Auch hier gilt, dass sich „Mitglieder“ der „Open Source Szene“ sich explizit von einer „Ripper Szene“ distanzieren.

8.3 Die „Open Source Szene“

Im Gegensatz zu den vorausgehenden „Szenen“ ist die „Open Source Szene“ völlig legal. Sie besteht in erster Linie aus Personen mit der festen Überzeugung, dass es für die zukünfigte und gegenwärtige Softwareentwicklung als positiv zu betrachten ist, wenn der Source Code der Software allen frei zur Verfügung steht. Sie glauben, dass Software prinzipiell Fehler enthält und dass es wahrhscheinlicher ist, das diese Fehler gefunden werden, wenn viele unabhängige Personen den Code lesen und verbesssern. Das Hauptkommunikationsmittel innerhalb der „Open Source Szene“ sind Internet Foren, in denen Programme vorgestellt, geplant und verbessert, Fragen gestellt und Hilfe angeboten wird. Besonders von Vorteil sind diese Foren meist für Programmierneulinge, die trotz weniger Erfahrung schnell und einfach Lösungen für Probleme erhalten können, aber auch erfahrenere Programmierer greifen oft auf diese Foren zu, um Mitarbeiter für große Projekte, Tester, neue Ideen oder andere Lösungen zu finden. Oft wird diesen Foren, wie auch der gesamten „Open Source Szene“, vorgeworfen eine Form des digitalen Kommunismus zu sein, da alles auf einer freiwilligen Basis geschieht und dementsprechend umsonst ist, doch genau dies ist es was diese „Szene“ für viele anziehend macht.

8.4 Persönliches Kommentar

Persönlich haben wir einen sehr positiven Eindruck der „Open Source Szene“ und seiner Mitglieder bekommen, da meist schnell, freundlich und richtig auf Fragen oder Anregungen geantwortet wird. Aber leider gibt es auch hier „schwarze Schafe“, die speziell Neulinge unfreundlich behandeln.

9 Gefahren für Open Source

Es gibt mehrere Entwicklungen die von Personen aus der Open Source Szene als Gefahr für die weitere Entwicklung des Open Source bezeichnet werden. Hier soll eine Auswahl der wichtigsten dieser Gefahren gezeigt werden.

9.1 Missachtung der Lizenzen

Ein weiteres großes und sehr aktuelles Problem für Open Source ist die Missachtung der Lizenzen. Das heißt, dass Konzerne Teile von Open Source Programmen oder Libraries in ihre eigenen Programme integrieren ohne dieses danach als Open Source weiterzugeben. Das ist zwar illegal, zumindest wenn die „geklauten“ Teile unter der GNU GPL standen, aber der einzige Beweis dafür stellt der nicht zugänglichen Code der Firmen dar. Und um an diesen Code heranzukommen bedarf es eines richterlichen Beschlusses, welcher nicht ohne Beweise zu bekommen ist.

9.2 Missbrauch der Lizenzen

Auch der Missbrauch der Lizenzen durch große Firmen ist ein großes Problem für die Open Source Initiativen. Hierbei versuchen Firmen beispielsweise Programme in mehrere Unterprogramme zu unterteilen, die einzeln zwar unbrauchbar sind, aber alle gemeinsam gut vermarktet werden können. Die Unterprogramme werden dabei so gebildet, dass man zum Beispiel ein Unterprogramm hat, das Open Source Code und Libraries verwendet und demzufolge auch unter die Lesser GPL (oder gar die GPL) gestellt wird, der Code der anderen Unterprogramme bleibt jedoch versteckt und geheim. Somit kann ein Programm (bestehend aus „unabhängigen“ Unterprogrammen) so vermarktet werden, dass nur unnütze Teile von ihm weitergegeben werden dürfen und das obwohl diese Firmen auf Open Source Code zugegriffen haben. Auf diese Weise umgehen die Firmen die Lizenzen ohne sich illegal oder ihre Programme zu Open Source zu machen. Dieses Verfahren ist jedoch in einem rechtlich noch ungeklärten Graubereich und ist moralisch schwer bedenklich.

9.3 Patentwesen in Europa

Es gibt in Europa Bewegungen mit dem Ziel sogenannte „Software Patente“ zu ermöglichen, obwohl diese „Software Patente“ keine Erfindungen im eigentlichen Sinn, sondern Logik, also logische Lösungen für Probleme, darstellen. Per Gesetz und Definition sind allerdings nur Erfindungen im klassischen Sinn patentierungswürdig. Sollte es diesen Bewegungen, die vor allem von großen Software–Firmen — wie zum Beispiel Microsoft — vorangetrieben werden, gelingen diese neue Art der Patente durchzusetzen, so hätte das neben wichtigen wirtschaftlichen Folgen auch schwere Konsequenzen für die Open Source Bewegung. Wenn es Microsoft gelingen sollte, in Bezug auf das Windows Betriebssystem hunderte kleine Software–Patente an sich zu binden, wäre es faktisch unmöglich, GNU/Linux als solches ganz aufrechtzuerhalten. Das liegt daran, dass es für Logik–Probleme manchesmal nur eine oder wenige logische Lösungen gibt — patentiert man die, bleibt der Weg für andere versperrt, selbst falls diese einen anderen programmiertechnischen Weg gehen sollten.

Aber nicht nur für die Open Source Bewegung könnten solche Patente fatale Folgen haben, auch für die „normale“ Wirtschaft, könnten sie verherrend sein: Da es Logiken für die Software Lösung eines solchen Problems im Gegensatz zu technischen Problemlösungen nicht in beliebiger Anzahl gibt, könnten undurchbrechbare Monopole entstehen. Patente auf Logiken könnten deren weitere Entwicklung auf Jahrzehnte verhindern, oder zumindest stark verlangsamen. Dies könnte besonders bei dem noch enormen Tempo der Entwicklungs der Software–Industrie stark negative Auswirkungen haben. Auch würden solche Patente auf Logiken die Software stark verteuern, da ihre Entwicklung erheblich mehr Kosten verursachen würde, weil jedesmal alle benutzten Logiken[23] aufwendig darauf geprüft werden müssten, ob sie vielleicht schon patentiert sind. Das würde de facto auch freie Software unmöglich machen.

[23] auch die ganz kleinen

9.4 Vorgehen durch Microsoft

Es gilt als ein offenes Geheimnis, dass Microsoft Softwareentwickler wie auch Hardwarehersteller, die offen Linux unterstützen, benachteiligt oder einfach nicht im selben Maß unterstützt wie Unternehmen, die nur für Windows entwickeln. Dieses Vorgehen erinnert stark an das Vorgehen Microsofts als es um die Ablösung des ISA–Ports durch PCI–Ports ging. Damals hat Microsoft allen Hardwareherstellern, die ISA–Ports unterstützen wollten den Entzug der Rechte am Windows–Logo angedroht. Bei dem monopolartigen Marktanteil von Microsofts Windows im Bereich der Home PCs bedeutete und bedeutet dies immer noch schwere Einbußen oder sogar das absolute Aus für diese Firmen. Mit Hilfe dieser Benachteilung von Firmen, die Linux unterstützen, versucht Microsoft ein weiteres Verbreiten von Linux auf dem Home–PC–Bereich einzuschränken. Als populäres Beispiel für dieses Vorgehen, wollen wir die Sonderfonds benutzen:

Nach einer firmeninternen Mail von Microsoft–Vertriebschef Orlando Ayala wurden diese Sonderfonds eingerichtet, um das Scheitern großer Geschäfte durch aggressive Preisnachlasse zu verhindern.[9]

9.5 TCPA, TCG

Trusted **C**omputing **P**latform **A**lliance oder wie ihr neuer Name ist **T**rusted **C**omputing **G**roup ist ein Zusammenschluss der größten Hard– und Softwareproduzenten dieser Welt. Mit Hilfe des **T**rusted **P**latform **M**odule, einem Chip der auf jegliche Hardware integriert werden soll, will dies Gruppe versuchen den Computer sicherer und damit „vertrauenswürdiger“ zu machen. Für viele Computerspezialisten stellt sich jedoch die Frage, wer hier wem vertrauenswürdiger gemacht werden soll. Mit Hilfe des TPM ist es möglich zu kontrollieren, welches Betriebssystem gestartet wird und somit beispielsweise Linux zu unterbinden. In Zusammenarbeit mit einem entsprechend ausgestatteten Betriebssystem, wie zum Beispiel dem neuen geplanten Windows Codename Longhorn, wäre dieser Chip de facto auch fähig zu kontrollieren welche Software auf diesem PC läuft. Da einer der Hauptkonzerne der TCPA und der TCG Microsoft ist, vermuten viele Open Source Anhänger, dass der TPM–Chip in erster Linie ein Monopol durch Microsoft und andere Mitglieder der TCPA/TCG bringen soll, indem er jeglichen Einsatz von Open Source verhindert.

10 Zukunft

Es gibt viele verschiedene Ansichten darüber, was mit Open Source in Zukunft geschehen wird.

Manche Firmen und (aus ihrer Sicht) pessimistisch eingestellte Open Source Programmierer gehen davon aus, dass die oben vorgestellten Gefahren direkt oder indirekt in naher Zukunft zu einem starken Rückgang der Verbreitung von Open Source führen wird. Andere meinen, dass dies nicht auf Grund der „künstlich erzeugten“ Gefahren geschehen wird, sondern weil es immer weniger Programmierer geben wird, die bereit sind auf freiwilliger Basis und damit unentgeltilich zu arbeiten. Sie sind also der Meinung, dass der Open Source Gedanke auf Grund von Programmierer–Mangel aufgegeben werden muss.

Aber es gibt auch andere Ansichten über die Zukunft des Open Source. So geht die Open Source Initiative davon aus, dass nach und nach immer mehr Programmierer zu Open Source „wechseln“. Sie geht auch davon aus, dass immer mehr Firmen bereit sein werden, in Open Source zu investieren und erwartet, dass sich auch rießige Konzerne der Open Source Bewegung anschließen werden. Beide Trends sind zumindest im Moment wirklich zu beobachten, beispielsweise investiert IBM[24] jährlich Millionen für die Unterstützung von Open Source. Als übermässig optimistisch kann also nur die weitere Hoffnung dieser Initiative angesehen werden, dass Open Source sich als Standard durchsetzen wird.

Wir persönlich vermuten auf Grund unserer Beobachtungen der akuten Entwicklungen, dass die Wahrheit wie so oft irgendwo in der Mitte liegt. Das heißt, dass Open Source nicht „aussterben“ wird, das gleiche aber auch für kommerzielle Software zutrifft. Wir glauben, dass beide Entwicklungsarten ein Recht auf Existenz und Anwendung haben und in einer koexistenten Form weitergeführt werden sollten.

[24] eine Firma, die sicher mit Recht als rießiger Konzern zu betrachten ist

A Begriffe

A.1 Betriebssystem

Grundbefehlsverarbeitungs–Programm eines Computers; bei den meisten eine Windows–Version

A.2 Kernel

Kern des *Betriebssystems*, welcher die Grundbefehle, wie zum Beispiel Hardware– und Datenzugriff, enthält.

A.3 Unix

Unix ist eines der ersten *Betriebssysteme*. Es unterstützt die Benutzung durch unterschiedliche Anwender, welche unterschiedliche Rechte besitzen können.

A.4 Linux

Linus' Uni**x** — ein Open Source *Kernel*, der auf *Unix* basiert. Heute versteht man unter Linux meist GNU/Linux, ein gesamtes *Betriebssystem*, das aus dem Linux–Kernel und dem *GNU* Softwarepaket besteht, und eine Konkurrenz zu Microsoft Windows darstellt.

A.5 Tux

Ein Pinguin, der das Markenzeichen und Maskottchen der *Linux*gemeinde ist.

A.6 Software

Alles was auf einem Rechner ausgeführt wird, bezeichnet man als Software; so zum Beispiel: Spiele, Programme und *Betriebssysteme*

A.7 Source Code

Source Code, zu deutsch Quellcode, ist klar lesbarer Text, der *kompiliert* und *gelinkt Software* ergibt.

A.8 Kompilieren

Vorgang bei dem aus dem klar lesbaren *Source Code* Maschinencode wird.

A.9 Library

Sammlung *kompilierter* Funktionen und Befehle für Programmierer.

A.10 Linken

Vorgang bei dem aus mehreren bereits *kompilierten* Codefragmenten oder *Libraries* ein Programm oder eine neue *Library* entsteht.

A.11 GNU

GNU **N**ot **U**nix — Abkürzung für das erste wieder freie Programmpaket.

A.12 Proprietär

In der Justiz ist der Begriff „proprietär“ gleichbedeutend mit „urheberrechtlich geschützt“. Davon abweichend wird der Begriff von der Open Source Gemeinde für *Software*, Standards, Dateiformate, Protokolle usw. benutzt, die nicht „frei“ sind. „Proprietäre *Software*“ ist also nicht–„freie“ *Software*.

A.13 EULA

End **U**ser **L**icense **A**greement — Vertragsähnliche Vereinbarung zwischen dem Hersteller einer *Software* und ihrem Benutzer. Hat in der EU nicht den Stellenwert eines juristischen Vertrages, in den USA schon.

A.14 Remote–Controll–Tools

Kleine Programme, deren Sinn darin besteht, einen Computer über ein Netzwerk von einem zweiten Computer aus zu bedienen.

A.15 Server & Clients

Rechnerbezeichnung in der Netzwerkverwaltung:

Client: untergeordneter, sich am Server anmeldender Computer

Server: übergeordneter Computer, der Informationen für Clients, die sich mit ihm verbunden haben, bereitstellt.

A.16 Active–X

Funktions– und Befehlspaket von Microsoft, das Programmieren dabei helfen soll fast unabhängige Programme in ein anderes einzubetten.

B Quellenangaben

Literatur

[1] http://www.bundestux.de

[2] http://www.cowo.de

[3] http://www.freesoftware.org

[4] http://www.fsf.org

[5] http://www.ftd.de

[6] http://www.gnu.org/licenses/fdl.html

[7] http://www.gnu.org/licenses/gpl.html

[8] http://www.gnu.org/licenses/lgpl.html

[9] http://www.heise.de

[10] http://www.heise.de/security/dienste/browsercheck

[11] http://idn.isc.org

[12] http://www.linux-muenchen.de

[13] http://www.macguardians.de

[14] http://www.muenchen.de

[15] http://www.opensource.org

[16] http://www.opensource.org/docs/definition.php

[17] http://www.pro-linux.de

[18] http://www.suse.de

[19] http://www.theregister.co.uk

[20] Ten Myths about Open Source Software, Tim O'Reilly, 11/01/1999

[21] Magazin c't Ausgabe 10 des Jahres 2004 herausgeben von Heise

[22] iX Nr.11, Jahrgang 2000, Seite 10 und folgende

[23] Antrag der ÖDP–Stadträtin Mechthild v. Walter vom 16.04.2003

[24] Presseerklärung der Münchner–SPD vom 26.04.2003

[25] Presseerklärung der Münchner–SPD vom 13.06.2003

[26] Presseerklärung der Münchner–SPD vom 02.12.2003

[27] Presseerklärung der Münchner–GRÜNEN vom 26.04.2003

[28] Presseerklärung der Münchner–GRÜNEN vom 25.05.2004

[29] Microsoft Windows XP EULA

C Unsere kleine Umfrage bei Schülern

Unsere kleine Umfrage an der Oskar–Schwenk Schule Waldenbuch[25] brachte folgende Erkenntnisse zu Tage: Linux lag vor allem in der Einschätzung der Stabilität vor Windows (1,7 zu 3,1[26]), wohingegen bei der Sicherheit Linux nur noch knapp vor Windows lag (2,1 zu 2,4). Interessanterweise wird Linux mit 1,8 Notenpunkten Unterschied als deutlich komplexer angesehen[27], aber viele stellen sich den Wartungsaufwand deutlich geringer vor (2,0 zu 3,1). Auf die Frage, ob man sich die private Nutzung eines Linuxsystems vorstellen könne, war die häufigste Antwort: „Nein, das ist viel zu kompliziert!“ Diejenigen, die sich selbst ein Linuxsystem zutrauen würden, lehnten dieses aber meisten aus dem Grund ab, man könne nicht so viele Spiele ohne größere Anstrengungen auf diesem System spielen, was zweifelsohne der Wahrheit entspricht, weil viele Spielehersteller ihre Spiele nur für Windows bzw. Mac ausliefern. Immerhin 7% haben bei sich zu Hause ein Linux–System (meistens SUSE).

[25] Nur 45 Befragte, d.h. sie kann nicht als repräsentativ angenommen werden (was sie ohnehin nicht kann, weil es größtenteils Schüler sind)

[26] Schulnoten

[27] es sei dahingestellt, ob Komplexität positiv oder negativ zu werten sein

D GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document „free“ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of „copyleft“, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **„Document“**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **„you“**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **„Modified Version“** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **„Secondary Section“** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **„Invariant Sections“** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **„Cover Texts“** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **„Transparent“** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not „Transparent“ is called **„Opaque“**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **„Title Page“** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, „Title Page“ means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **„Entitled XYZ“** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **„Acknowledgements“**, **„Dedications“**, **„Endorsements“**, or **„History“**.) To **„Preserve the Title“** of such a section when you modify the Document means that it remains a section „Entitled XYZ“ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled „History“, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled „History“ in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the „History“ section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled „Acknowledgements“ or „Dedications“, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled „Endorsements“. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled „Endorsements“ or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled „Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled „History“ in the various original documents, forming one section Entitled „History“; likewise combine any sections Entitled „Acknowledgements“, and any sections Entitled „Dedications“. You must delete all sections Entitled „Endorsements“.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an „aggregate“ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled „Acknowledgements“, „Dedications“, or „History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License „or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled „GNU Free Documentation License“.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the „with...Texts.“ line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.