FAKULTÄT FÜR !NFORMATIK

universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

Free Open Source Software

Freie Software an allgemeinbildenden höheren Schulen

Verfasser

Thomas Michael Weissel

angestrebter akademischer Grad

Magister der Naturwissenschaften (Mag. rer.nat)

Wien, 2009

Studienkennzahl lt. Studienblatt: A190 884 299
Studienrichtung lt. Studienblatt: Lehramt Informatik und Informatik Management
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn Gerald Futschek

Free Open Source Software

Freie Software an allgemeinbildenden höheren Schulen

"Der fundamentale Akt von Freundschaft unter denkenden Wesen besteht darin, einander etwas beizubringen und Wissen gemeinsam zu nutzen. Dies ist nicht nur ein nützlicher Akt, sondern es hilft die Bande des guten Willens zu verstärken, die die Grundlage der Gesellschaft bilden und diese von der Wildnis unterscheidet.
Dieser gute Wille, die Bereitschaft unserem Nächsten zu helfen, ist genau das, was die Gesellschaft zusammenhält und was sie lebenswert macht. Jede Politik oder jedes Rechtssystem, das diese Art der Kooperation verurteilt oder verbietet, verseucht die wichtigste Ressource der Gesellschaft.
Es ist keine materielle Ressource, aber es ist dennoch eine äußerst wichtige Ressource."
[GRASS, S.224]

Inhaltsverzeichnis

1 Übersicht

1.1 Kurzfassung

In dieser Arbeit wird anhand ausgewählter Texte und Studien erörtert, ob bzw. weshalb, in allgemeinbildenden höheren Schulen, freie Software eingesetzt werden sollte. Erweitert wird die Argumentation durch eine Reihe von Interviews und praktischen Beispielen, welche die Behauptung untermauern, dass der Einsatz von freier Software für Schulen, insbesondere für den Unterricht von Vorteil ist und sowohl die wirtschaftliche Abhängigkeit im IT Bereich zu reduzieren vermag als auch die Kosten senken kann. Die Untersuchung des Themenfeldes hat nicht nur die These bestätigt sondern auch ergeben, dass freie Software in Schulen weiterführend eine Qualitätssteigerung des Lehrangebotes bedeuten kann, eine erfolgreiche Migration jedoch von einer Vielzahl an Variablen abhängt und daher jede Situation für sich genommen geprüft werden muss und keine pauschale Aussage über die Erfolgschancen einer solchen Umstellung getroffen werden kann. Abschließend wird aufgezeigt, welche Relevanz freie Software in Zukunft haben wird und warum Linux, als wichtigster Vertreter von freier Software schon jetzt in viele Lebensbereiche Einzug hält und aus diesen nicht mehr wegzudenken ist.

In this work, the question as to whether free software should be used in secondary schools is discussed. The argument is advanced through a series of interviews and practical examples which support the assertion that the use of free software for schools, especially for teaching, is beneficial and both the economic dependence on the IT field as well as the costs can be reduced. The investigation of the topic has not only confirmed the theory, but also revealed that free software in schools can lead to an improvement in quality of the courses and also that a successful migration depends on a multitude of variables. Therefore, each situation for itself should be examined and no blanket statement on their chances of success of a migration can be taken. Finally, attention has been drawn to what relevance free software will have in the future and, furthermore, why Linux, as the main representatives of free software, is already present in many areas of life and why those areas cannot be contemplated without it anymore.

1.2 These

Der Einsatz von freier Software in Bildungseinrichtungen ist durch die Sicherstellung von Interoperabilität und uneingeschränktem Gebrauch im täglichen Lehr- und Lernszenario sowie wegen der Wiederverwertbarkeit und der legalen Weitergabe eingesetzter Software an Schüler, Eltern und Lehrer nicht nur wegen des ideologischen und lizenzrechtlichen Hintergrunds vorteilhaft für Schulen, sondern ermöglicht es zudem, die wirtschaftliche Abhängigkeit im IT Bereich zu reduzieren und Kosten zu sparen.

1.3 Aufgabenstellung

Ziel der vorliegenden Arbeit ist es, anhand ausgewählter Texte und Studien zu erörtern und zu überprüfen ob, respektive zu begründen, weshalb in allgemeinbildenden höheren Schulen, freie Software anstelle von proprietärer Software eingesetzt werden sollte. Einzelne Aspekte - verringerte wirtschaftliche Abhängigkeit, finanzieller Hintergrund, offene Schnittstellen und Wiederverwertbarkeit - werden im Zuge der Aufarbeitung meiner These speziell auf ihre Alltagstauglichkeit hin untersucht. Die getroffenen Aussagen wären zu großen Teilen auch übertragbar auf ähnliche Institutionen im öffentlichen Raum. Da eine Betrachtung von Bildungseinrichtungen im allgemeinen jedoch den gesteckten Rahmen sprengen würde, liegt der Fokus dieser Arbeit bei allgemeinbildenden höheren Schulen, was darüber hinaus auch mit dem Ausbildungsziel meines Studiums korreliert.

1.4 Ergebnis

Im Gespräch mit Personen aus der Praxis über die Arbeit mit freier Software in der Schule ergibt sich eine erstaunliche Übereinstimmung der Ansichten und Erfahrungen sowohl unter den einzelnen Gesprächspartnern als auch mit den Studien, die zur Untersuchung herangezogen wurden. Durch die intensive Aufarbeitung des Themenfeldes ließ sich die These weitgehend bestätigen. Eine allgemeingültige Aussage über den Einsatz von Free Open Source Software an Schulen lässt sich allerdings nicht treffen, da der Erfolg von einer Vielzahl an Variablen abhängt und jede Ausgangssituation für sich genommen geprüft werden muss. Freie Software wird jedoch in Zukunft ein Faktor sein, mit dem man rechnen muss.

2 Vorgehensweise

2.1 Persönlicher Zugang zum Thema

Mein Wissensdurst in technischen Belangen wurde schon zu Beginn meines Informatikstudiums während einer Lehrveranstaltung nicht etwa durch die Beantwortung all meiner Fragen, sondern vielmehr durch das Aufzeigen der unendlichen Bandbreite, die das weite Gebiet der Informatik mit sich bringt, zufrieden gestellt. Durch die Vermittlung des Gefühls, hier unbegrenzt neue Fragen stellen zu können und zugleich auch der Gewissheit, zunächst alle Antworten darauf finden zu können, begeisterte ich mich schnell für die Idee von Open Source. Hier würde man die optimale Umgebung finden, um sich tief in die Materie einzuarbeiten und sich einem endlosen Lernprozess in freier Interaktion mit anderen hinzugeben; Systeme zu studieren und bis zu kleinsten Details zu verstehen. Welche Rahmenbedingungen von Free Open Source Software dies u.a. ermöglichen und was diese freie Software genau darstellt, werde ich im Folgenden speziell im Kapitel „Grundlagen“ erörtern.

Die prägende Lehrveranstaltung hieß „Technische Praxis der Computersysteme“ bei Mag. Florian Wisser. Im Zuge dieser Lehrveranstaltung sollten wir Näheres über die Funktionsweise von Betriebssystemen lernen. Die Entscheidung fiel unter anderem auf Grund einer gewissen Unmöglichkeit, diverse Vorgänge der Computerpraxis am marktbeherrschenden Betriebssystem einfach zu veranschaulichen, auf das freie Betriebssystem GNU/Linux, welches zu diesem Zeitpunkt als Redhat 7.3 auch auf der Universität Wien eingesetzt wurde. Der modulare, skriptbasierende Bootvorgang, die lesbaren Konfigurationsdateien, die vom System getrennte grafische Oberfläche und die freie, kostenlose Verfügbarkeit des Systems machen es zu einem idealen Vorzeigemodell, um die Funktionsweise eines Betriebssystems zu lehren und zu lernen. Des Weiteren bleibt zu erwähnen, dass ich im Privatbereich mehrheitlich mit freier Software arbeite, für diverse erfolgreiche Migrationen auf ein freies System verantwortlich bin und mit Systemadministration sowohl unter Linux als auch unter Windows vertraut bin. Als Autor der freien Content Management Lösung „typeX-press“ beschäftige ich mich mit der Materie tiefer gehend und kann auf etwa 5 Jahre einschlägige Erfahrung mit freier Software zurückgreifen.

2.2 Aufbau der Arbeit

Zu Beginn wird im Kapitel „Grundlagen“ auf die Open Source Szene, dessen Basis, die GNU General Public Licence, sowie auf wichtige Personen und Organisationen genauer eingegangen, um die Wissensbasis für die nachfolgende Darlegung zu schaffen. Im Kapitel „FOSS im Schulbetrieb“ werden relevante Themenbereiche für Schulen ausgerollt sowie anschließend das Modellbeispiel „BG Rechte Kremszeile“ vorgestellt, welches die Theorie dieser Arbeit in die Praxis überführen soll. Abschließend wird das Open Source Projekt „typeX-press“ und dessen Einsatzmöglichkeiten in Schulen besprochen und versucht, einen kurzen Überblick zu geben, wie freie Software auf viele Bereiche der modernen Welt Einfluss hat, weshalb sie aus dieser nicht mehr wegzudenken ist und welche Relevanz freie Software in Zukunft haben wird.

2.3 Wissenschaftliche Methodik

Diese vorwiegend theoretische Arbeit beruft sich auf aktuelle Studien und Texte, die sich mit der Relevanz von Free Open Source Software (FOSS) in öffentlichen Einrichtungen und ihrer Bedeutung für Demokratie, Wissensfreiheit und Zusammenarbeit beschäftigen. Zur besseren Übersicht werden zitierte Texte optisch hervorgehoben und der Quellennachweis durch ein System aus erdachten Abkürzungen vereinheitlicht, welche sich als Verweise auf das alphabetisch geordnete Quellenverzeichnis verstehen. Diese Kürzel finden sich stets am Ende eines Zitats und werden, falls notwendig, durch Zusätze zur genaueren Definition der Quelle ergänzt.

Der Bezug zur Praxis wird durch die Auswertung qualitativer Interviews sowie durch Beispiele für den Einsatz von FOSS im Unterricht hergestellt. Anhand des Beispieles „BG Rechte Kremszeile“ und des Projekts „typeX-press“ zeige ich, wie der Einsatz von freier Software in der Realität funktionieren kann. Bei der Durchführung der Interviews habe ich mich für die leitfadengestützte Methode offener Interviews, orientiert an den Arbeiten von Michael Meuser und Ulrike Nagel, entschieden. Diese Methode ermöglichte es mir, den Rahmen des Interviews abzustecken und darüber hinaus das Gespräch für neue Aspekte seitens der Interviewpartner offen zu halten. Die Ausarbeitung des Interviewleitfadens erwies sich komplizierter als anfangs angenommen. Testinterviews mit Kollegen konzentrierten meine Aufmerksamkeit wiederholt auf die Frage, inwiefern mir die Interviewfragen ein tieferes Verständnis der Materie ermöglichen könnten. Mit Hilfe des Interviewleitfadens, nachzulesen im „Anhang“, wurden ausgewählte IT-Fachkräfte befragt und die Ergebnisse dieser Gespräche wurden in meine Argumentation aufgenommen. Die Gesprächspartner wurden von mir sorgfältig in Hinblick auf einschlägige Erfahrung mit freier Software im Bildungsbereich ausgesucht. In meiner empirischen Untersuchung habe ich mich somit auf Schulen konzentriert, die zumindest teilweise Free Open Source Software einsetzen, da ich nach eingehender Überlegung zu dem Schluss kam, nichts über das Thema dieser Arbeit von Institutionen, die sich mit diesem noch nicht auseinandergesetzt haben, lernen zu können bzw. es meine persönlichen Möglichkeiten übersteigen würde, jene Schulen zu finden, die eine Auseinandersetzung mit dem Thema „freie Software“ erfahren, sich aber dennoch für eine ausschließlich proprietäre Lösung entschieden haben; vorausgesetzt solche existierten (was ohne Zweifel sehr interessant wäre).

Während der Interviews musste ich zu meiner Überraschung feststellen, dass sich nicht nur Inhalte, sondern teilweise auch konkrete Formulierungen wiederholten und unter anderem auch mit meinen eigenen Texten übereinstimmten. Es scheint ein allgemeiner Konsens zwischen der Theorie und den praktischen Erfahrungen zu geben. Die aufmerksamen Leser sind daher gebeten, dies zu berücksichtigen, wenn Formulierungen meiner Arbeit Ähnlichkeiten zu gegebenen Antworten aufweisen (nachzulesen im Anhang unter „Transkribierte Interviews"). Den Interviews zeitlich nachfolgende Parallelen in der Argumentation werden von mir stets als Zitate in den Fließtext eingebunden, um diese mit den Worten der Experten zum Ausdruck zu bringen.
Die Auswertung der Interviews erfolgte auf Basis der gestellten Leitfragen. Nach Meuser und Nagel ist eine partielle Auswertung von Experteninterviews, die als eines von mehreren Instrumenten zur Datenerhebung in einer Untersuchung eingesetzt werden, ausreichend. Auch verzichtete ich bei der Transkription darauf, nonverbale Elemente wie Pausen oder Stimmlagen zu kennzeichnen, da diese nicht Gegenstand der Interpretation sind. [vgl. *MENA*]

In der vorliegenden Arbeit werde ich zugunsten der besseren Lesbarkeit und nicht zuletzt, um extravagante Wortkreationen zu vermeiden, versuchen, stilistisch passende Begriffe zu verwenden wie „Anwendungssoftware" anstelle von „AnwenderInnensoftware" oder „Lehrpersonen" anstelle von „LehrerInnen" und mich im Zweifelsfalle auf die geläufige Auffassung beziehen, dass in der deutschen Sprache Genus und Sexus, das grammatikalische und das natürliche Geschlecht, von einander getrennt sind. Ich werde daher das verbreitete generische Maskulin verwenden in der Hoffnung, dass sich die Leserinnen unter dieser Voraussetzung genauso wenig um ihre weibliche Teilnahme betrogen fühlen bei Sätzen, in denen von „*den Schülern*" die Rede ist, wie auch das männliche Geschlecht implizit ist, wenn generische Feminina bzw. generische Neutra verwendet werden.

3 Existenzgrundlagen von freier Software

3.1 Ideologischer Hintergrund der freien Software

"Füge eine Kleinigkeit zur anderen, und das Ergebnis wird ein großer Haufen sein." (Ovid)

Der folgende Abschnitt widmet sich der Idee freier Software, erläutert die Bezeichnung und stellt sie in Gegensatz zu proprietärer Software.

Die Bezeichnung „Freie Software“ weist auf politische, soziale und individuelle Freiheit hin und unterscheidet sich damit in ihrer Bedeutung leicht von der ebenso gängigen Bezeichnung „Open Source Software“, welche in erster Linie für die technischen Vorteile von freier Software sensibilisieren möchte und daher oftmals den Zusatz „free“ erhält, um auf die damit verbundenen Freiheiten aufmerksam zu machen. In der vorliegenden Arbeit werden die Begriffe „freie Software“, „Free Open Source Software“ (FOSS) als auch „Open Source Software“ (OSS) von mir synonym verwendet. Es scheint zunächst wichtig, ein weit verbreitetes Missverständnis aufzuklären: Freie Software, wie Open Source Software oft genannt wird, bedeutet nicht, dass sie gezwungenermaßen kostenfrei sein muss. Obwohl FOSS zumeist auch gratis ist und ohne weiteres legal von diversen Spiegelservern (Mirrors) heruntergeladen werden kann, ist die Kostenfreiheit keine Voraussetzung und in keiner Definition zu finden. So ist es durchaus legitim, Kopien freier Software oder eventuelle mit dieser Software zusammenhängende Services auch kostenpflichtig zu gestalten; ein Umstand, auf den ich im Abschnitt „Business“ noch zu sprechen kommen werde.

Richard Stallman trifft es mit folgenden Worten auf den Punkt:

„free as in free speech, not as in free beer“ [Richard Stallman, > 40 000 Hits]

Als FOSS bezeichnet man jene Programme, deren Lizenzen ihren Benutzern die Freiheit zusichern, diese für ihre Zwecke zu benutzen, zu studieren oder zu verändern und Kopien davon weiterzugeben, sowohl von dem Original als auch von der von ihnen veränderten Version, ohne dabei Zahlungen an die bisherigen Entwickler leisten zu müssen.

Die Free Software Foundation, deren Organisation und Entstehung ich im Kapitel 3.3 vorstellen werde, hat folgende Definition von freier Software veröffentlicht:

- ***Die Freiheit, das Programm für jeden Zweck auszuführen.***
  Einschränkungen bezüglich der Verwendbarkeit von Software, bezogen auf Zeit ("30 Tage Testphase", "Lizenz endet am 1. Januar 2004"), Zweck ("Verwendung gestattet für Forschung und nicht-kommerzielle Anwendung", "darf nicht für Leistungsvergleiche (Benchmarking) eingesetzt werden") oder willkürliche geographische Beschränkungen ("darf nur im Land X verwendet werden") machen ein Programm unfrei.
- ***Die Freiheit, die Funktionsweise eines Programms zu untersuchen, und es an seine Bedürfnisse anzupassen.***
  Rechtliche oder praktische Einschränkungen der Einsicht in die Programmfunktion oder der Veränderbarkeit eines Programms, wie der zwingende Erwerb spezieller Lizenzen, die Unterzeichnung eines Stillschweigeabkommens oder - bei Programmiersprachen, die mehrere Formen der Repräsentation bieten - die Zurückhaltung der üblicherweise bevorzugten Bearbeitungsform ("Quellcode") machen es ebenfalls proprietär (unfrei). Ohne die Freiheit, ein Programm zu ändern, bleiben die Anwender vom Wohlwollen eines einzigen Anbieters abhängig.
- ***Die Freiheit, Kopien weiterzugeben und damit seinen Mitmenschen zu helfen.***
  Software kann praktisch ohne Kosten kopiert und weitergegeben werden. Das Verbot, ein Programm an eine Person weiterzugeben, die es braucht, macht dieses Programm unfrei. Die Weitergabe kann wahlweise auch gegen ein Entgelt erfolgen.
- ***Die Freiheit, ein Programm zu verbessern, und die Verbesserungen an die Öffentlichkeit weiterzugeben, sodass die gesamte Gesellschaft profitiert.***
  Nicht jeder ist in allen Bereichen ein guter Programmierer. Manche Leute können überhaupt nicht selbst programmieren. Diese Freiheit erlaubt jenen, die nicht die Zeit oder die Fähigkeit haben, ein Problem selbst zu lösen, indirekt von der Freiheit, ein Programm zu ändern, Gebrauch zu machen. Auch das kann gegen ein Entgelt geschehen.

[FSFE, Was ist freie Software]

Diese Freiheiten sind keine Pflichten, sondern Rechte. Jede Einzelperson kann frei wählen, auf diese zu verzichten oder sie anzunehmen.

Geht es nach Richard Stallman, so ist Software und dessen Quellcode wie Wissen zu behandeln und auf gleicher Ebene wie Bildung einzustufen und damit die Pflicht eines jeden sozialen Wesens, diese im Bedarfsfalle weiterzugeben und dieses „Wissen“ zu vermitteln. Ihm zufolge müssen Ideen frei sein, damit wir uns als Spezies weiter entwickeln können; die Arbeit und der Aufwand, um diese zu entwickeln oder weiterzugeben, sind jedoch nicht notwendigerweise kostenfrei. Würden wir, als Gedankenexperiment, Bildung wie herkömmlich lizenzierte, proprietäre Software handhaben, so dürfte kein Lehrer einem Schüler etwas beibringen, solange er nicht dem ursprünglichen Lehrer eine Abgabe für jede Unterrichtsstunde gezahlt hätte; niemand dürfte, frei nach Stallman, sein persönliches Wissen weitergeben, um z.B. seinem Nachbarn zu helfen, es sei denn, er habe sich dieses selbst erarbeitet. Oder anders umschrieben: Proprietäre Software verhält sich als Produkt wie ein Fahrzeug, dessen Motorhaube ausschließlich dem Hersteller Zugang zu der darunter liegenden Mechanik gewährt; ein Fahrzeug, dessen einzelne Komponenten nur der Hersteller wechseln kann. Auch im Falle eines Notfalles sichert ein gesetzliches Verbot diese Rechte des Herstellers und hindert den Besitzer daran. Software auf einer Ebene mit Wissen zu behandeln ist auch in der Open Source Szene durchaus ein kontroverses Statement. So haben auch die geführten Expertengespräche eine generell positive Haltung zur Offenheit des Quellcodes und der Möglichkeit der freien Weitergabe von Software zu Tage gebracht aus Gründen, auf die ich im Verlauf dieser Arbeit noch eingehen werde; jedoch die Software respektive den Programmcode einer Software direkt als Wissen, welches durch die Weitergabe vermittelt wird, zu bezeichnen, scheint in der Praxis eher eine unrealistische Idealvorstellung zu sein. So bemerkte Mag. Florian Wisser treffend:

> *„Natürlich gibt es im Programmcode immer wieder intelligente Lösungen zu finden. Wenn allerdings jetzt jemand dieses Wissen im wissenschaftlichen Bereich wirklich hergeben will, dann ist es typischerweise so, dass er das in ein kleines Paper mit Pseudocode schreibt, in dem es für einen User einfacher zu lesen, also es für einen Leser einfacher ist, die Ideen herauszuziehen als direkt aus dem Quellcode“ [Florian Wisser, Interview 2008]*

Während der Umweg über Pseudocode im universitären Bereich durchaus üblich ist, wird es dieser Code im Falle proprietärer Lösungen großer Softwarehersteller vergleichsweise kaum über den Firmensitz hinaus zu den Programmierern, die dieses Wissen benötigen könnten, schaffen. Dies liegt in der Natur proprietärer Lösungen in einer kapitalistischen Wettbewerbswirtschaft. Das Bundesministerium für Inneres definiert den Unterschied zwischen freier und proprietärer Software wie folgt:

> *Im Gegensatz zu Open Source Software ist proprietäre Software das Eigentum einer Person oder einer Organisation. In der Regel handelt es sich dabei um den Hersteller der Software. Die Nutzung der Software unterliegt den Lizenzbestimmungen, die der Eigentümer der Software festgelegt hat. Dabei ist ihre Vervielfältigung, Weiterverbreitung und Modifizierung meist untersagt. Unter der Voraussetzung, dass die entsprechenden Lizenzbedingungen akzeptiert werden, wird sie in einigen Fällen auch kostenlos angeboten. Diese Software ist jedoch keine Open Source Software. [BMI, Migrationsleitfaden, S.7]*

Heute scheint der Begriff „Open Source“ trotz seiner unklaren Definition etablierter. Im Laufe eines langen Diskurses, bei dem es unter anderem auch darum ging, freie Software, insbesondere die Idee von quelloffener Software für Firmen attraktiver zu machen, wurde von Eric Steven Raymond der Begriff „Open Source” aus Marketinggründen geprägt. Raymond war ein früher Mitarbeiter des GNU Projektes und gründete gemeinsam mit Bruce Perens im Jahr 1997 die Open Source Initiative (OSI).

Die OSI hat eine eigene ausführliche Definition von Open Source Software veröffentlicht.

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

[OSI, The Open Source Definition]

Die Open Source Definition der OSI versteht sich als Standard, an dem freie Lizenzen gemessen werden sollen.

Eine Copyleft-Lizenz, wie eine solche Lizenz oft genannt wird, soll also unter anderem die schnelle und einfache Weiterentwicklung von Software sicherstellen und daher die Freiheit, den unverschlüsselten Sourcecode zu lesen sowie diesen zu verändern, gewähren. Dies allein ist jedoch nicht genug, um unabhängige Softwareentwicklung und schnelles Voranschreiten von Verbesserungen in diesem Sektor zu garantieren, weshalb die uneingeschränkte Weitergabe von veränderter Software in die Definition mit aufgenommen wurde. Ein Ausschluss von Personen oder Personengruppen muss dezidiert untersagt sein, jeder soll am Entstehungsprozess teilhaben und die Software für seine Zwecke nutzen können, unabhängig von Herkunft und Hautfarbe. Der sechste Punkt der OSI-Definition versucht sicherzustellen, dass Open Source Software auch kommerziell genutzt werden kann. Wie bereits erwähnt ist diese Definition auch als Versuch zu verstehen, die Verwendung freier Software für Firmen attraktiver zu gestalten. Sogenannte „non-disclosure“ Vereinbarungen, die eine indirekte Restriktion bedeuten würden, und auch andere Lizenzfallen sollen durch diese vermieden werden und das Recht sichergestellt werden, mit der Software sowohl als Distributor als auch als Kunde machen zu können, was im eigenen Interesse liegt, solange die Rechte anderer dadurch nicht beschnitten werden.

Wie aus den Definitionen klar ersichtlich ist freie Software weit mehr als nur „Open Source Code”, welcher für jeden frei zugänglich ist. Microsofts CEO Steve Ballmer bezeichnete das GNU/Linux Betriebssystem (als Vorzeigebeispiel und Synonym für FOSS) als “kommunistisch”. Dies war freilich als Abwertung und Anlehnung an den erlebten „Realsozialismus“ gedacht, nicht an die Grundidee eines marxistischen Systems nach Karl Marx. Schon der frühere CEO Microsofts, Bill Gates, nutzte derartige Rhetorik, um die gesamte Bewegung zu diffamieren.

"There is a particular approach that breaks the cycle called the GPL that is not worth getting into today, but I don't think there is much awareness about how so-called free software foundations designed that to break that cycle. So free software was fine up until 1989, but then Vladimir Ilyich Stallman came up with the GNU GPL with the specific aim of breaking the virtuous cycle." [HMIOS]

In der Tat ist die freie Softwareszene eher weniger kommunistisch, doch in mancherlei Hinsicht lassen sich Marx' Ansichten vielleicht auf diese übertragen, selbst wenn die Communities hier geteilter Meinung sind. Durch das Internet sind die Produktionsmittel - ein Computer und ein Netzzugang - nicht mehr in der Hand einiger weniger, sondern für jeden zugänglich und einsetzbar. Dies machte die Entstehung von freier Software zu großen Teilen erst möglich und schaffte sogleich die Basis für die flächendeckende gerechte Verteilung der Software und vor allem der Entwicklungswerkzeuge, u.a. dem GNU-C-Complier. Die damit entwickelte Software wurde unter eine freie Lizenz gestellt, welche alle Rechte an die Benutzer weitergab. Das erstellte Produkt gehörte somit der Gemeinschaft und jeder war aufgerufen, zum Allgemeinwohl daran mitzuarbeiten, es zu verbessern und es zu verbreiten.

3.2 Wichtige Persönlichkeiten

3.2.1 Richard Stallman

Abb. 1: Richard (Che) Stallman

Richard Matthew Stallman ist Aktivist, Softwareentwickler und Gründer der Free Software Foundation im Jahr 1985. Er promovierte 1974 an der Harvard Universiy in Physik, während er bereits am Massachusetts Institute of Technology (MIT) angestellt war und dort u.a. am ersten erweiterbaren *Emacs* Texteditor und an einem Betriebssystem arbeitete. Im Jänner 1984 kündigte er am MIT, um sich dem ein Jahr zuvor gegründeten GNU Projekt zu widmen, dessen Ziel es ist, ein vollkommen freies Betriebssystem zu entwickeln. Dies war der Start der „Free Software Bewegung“. Seit den 90er-Jahren verbreitet Stallman die Ideen der Bewegung und tritt gegen die Erweiterung von Copyright Bestimmungen und Softwarepatenten öffentlich auf. Er schuf eine Lizenz, die unter dem Namen „GNU General Public License“ (GPL) bekannt wurde, welche den Anwendern erstmals die Rechte über ihre Software garantierte und sicherstellte, dass diese auch in Zukunft und für sämtliche Veränderungen und Erweiterungen dieser Software ebenfalls gelten würden. Es entstand das sogenannte Copyleft-Prinzip. Stallman hält Vorträge über GNU, die Free Software Bewegung und verwandte Themen wie "The Dangers of Software Patents", "Copyright and Community in the Age of the Computer Networks" und „The GNU General Public License v.3“. In all seinen Vorträgen vertritt Richard Stallman die Auffassung, dass die Idee von Copyrights und proprietärer Software unsozial und damit falsch sei. Er beschreibt ein System, welches nicht nur den freien Austausch von Software, sondern auch Ideen unterbindet und die Anwender in ihren Möglichkeiten einschränkt und positioniert sich gegen dieses in seinen Vorträgen.

3.2.2 Linus Torvalds

"Really, I'm not out to destroy Microsoft. That will just be a completely unintentional side effect." [QFLT, S.2]

Abb. 2: Linus (Tux) Torvalds

Nach dem Kauf seines ersten i386er-PCs, auf welchem Linus Torvalds mit „Minix“, einem UNIX Klon für diese Architektur, arbeitete, erkannte er schnell, dass er mit dessen Funktionalität nicht sehr zufrieden war. Aus lizenzrechtlichen Gründen war an eine Veränderung bzw. Erweiterung von Minix nicht zu denken. Zudem benötigte er für seine Zwecke ein Programm, das der „terminal emulation“ fähig war und er begann ein solches zu schreiben. Schnell entwickelte sich dieses Programm und wurde von ihm um Funktionen des Datentransfers und der Sicherung von Daten erweitert. Dies war die Geburtsstunde des ersten, freien Betriebssystemkernels. Linus Torvalds wollte seine Kreation „Freax“ nennen, entschied sich auf Anraten eines Freundes jedoch für die Namenskreation „Linux“. Im August 1991 verkündete er im Usenet, dass er an einem Betriebssystem arbeitete und wollte sich auf diese Weise Feedback und neue Ideen für sein System holen.

"From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(."
[USENET]

Linus Torvalds wurde am 28. Dezember 1969 in Helsinki, Finnland geboren und studierte an der Universität Helsinki „Computer Science“. Seitdem arbeitet er an Linux und koordiniert auch heute noch dessen Entwicklung.

3.3 Free Software Foundation

Der Zugang zu Software entscheidet, wer an der digitalen Gesellschaft teilnehmen kann. Um gleichberechtigten Zugang, Teilnahme und Wettbewerb aller Menschen im Informationszeitalter sicherzustellen, setzt sich die Free Software Foundation für digitale Freiheit in der Form von freier Software ein.

> *„Kein Mensch sollte jemals in einer Situation sein, in der er Software einsetzen muss, die ihm nicht die Freiheit der Nutzung, des Studiums, der Modifikation und der Weitergabe ermöglicht.“ [FSFE, „Helfen sie mit!“]*

Die 1985 von Richard Stallman gegründete „non governmental organisation“ FSF vertritt weltweit die Rechte aller Benutzer freier Software und fördert deren Verbreitung. Sie finanziert sich u.a. durch Spenden, erzielt aber große Teile ihres Einkommens aus Gebühren für Kopien freier Software und verwandten Dienstleistungen wie beispielsweise gedruckten Anleitungen und Dokumentationen zu freier Software sowie fertigen Distributionen. Durch diverse Kampagnen wie „Speak out against ACTA“ (Anti-Counterfeiting Trade Agreement), „BadVista“ und „End Software Patents “ macht die FSF permanent auf sich aufmerksam, setzt sich für die Verbreitung von „Ogg/Vorbis“ und „Open Document“ als Vertreter von offenen Standards ein. Sie fördert nicht nur die Verbreitung freier Software, sondern auch freier Hardware und veröffentlicht regelmäßig eine Liste von Softwareprojekten, denen man eine besondere Priorität für den Erfolg von freier Software zuspricht, als Empfehlung für freie Entwickler, um auf die Relevanz dieser Projekte hinzuweisen und Entwickler zur Mithilfe zu bewegen. So schreibt die FSF:

> *„Our list helps guide volunteers and supporters to projects where their skills can be utilized, whether they be in coding, graphic design, writing, or activism. We hope that you can find a project here where your skill, energy, and time can be put to good use.“ [FSF]*

Am 10. März 2001 bekam die Free Software Foundation Unterstützung in Form eines europäischen Pendants, der Free Software Foundation Europe (FSFE). Die FSFE nimmt sich der europäischen Aspekte von freier Software an, hat eine Anlaufstelle für Politiker, Anwälte und Journalisten ins Leben gerufen, um die rechtliche, politische und soziale Zukunft von Free Open Source Software zu sichern und ist in einem starken Netzwerk gleichgesinnter, auf der ganzen Welt verteilter Organisationen aktiv und hat Teams und Unterstützer in vielen europäischen Ländern.

Im Jahr 2005 hat eine Gruppe von Free-Software-Aktivisten in Lateinamerika öffentlich ihre Absicht erklärt, sich dem weltweiten Netzwerk der Free Software Foundations anzuschließen. Die FSFLA (Free Software Foundation Latinoamérica) ist somit neben der FSF, der FSFE und der FSFI (Free Software Foundation India) die vierte ihrer Art und setzte sich wie ihre Schwesterorganisationen das Ziel der Förderung, Bekanntmachung und Verteidigung freier Software und der Zusammenarbeit mit den anderen FSFs.

Geistiger Wohlstand statt geistigem Eigentum lautet die Devise der FSFE. Die Weltorganisation für geistiges Eigentum (WIPO, World Intellectual Property Organisation) hat bislang 23 internationale Verträge zu Patenten, Urheberrechten und Warenzeichen auf Basis des öffentlich-politischen Dialogs geschlossen. Die FSFE lehnt diesen generellen Trend zum Ausbau von Monopolen und Kontrolle entschieden ab und empfiehlt gemeinsam mit anderen Organisationen der Zivilgesellschaft den Umbau der WIPO zu einer Weltorganisation für geistigen Wohlstand (WIWO, World Intellectual Wealth Organisation), die sich der Forschung und der Verbreitung von Wissen verpflichtet fühlen soll. Nach Ansicht der FSFE sollten künstliche Monopole nur dort geschaffen werden, wo sie einen Nutzen für die Gesellschaft bringen, welcher neutral untersucht und nicht ungeprüft vorausgesetzt werden darf.

3.4 GNU/Linux

„The fundamental instance of Free Software is the GNU operating system, which is the only operating system that was ever developed for ethical reasons, instead of for commercial or technical reasons, which is why most systems were developed.“ [WIFS]

Richard Stallman begann 1984 mit der Entwicklung des GNU Betriebssystems, welches jedem Benutzer einen von Restriktionen befreiten Einsatz des Computers ermöglichen sollte, frei von wirtschaftlichen Interessen und der Kontrolle der Softwarefirmen und Entwicklern. Der Name GNU ist ein rekursives Akronym (**G**NU is **n**ot **U**nix) und soll auf die Ähnlichkeit des Systems zu Unix hinweisen. Erst 1992 wurde die letzte große Lücke in GNU geschlossen, als Linus Torvalds den Betriebssystemkern „Linux“ veröffentlichte. Durch die Kombination aus GNU und Linux war das erste komplett freie Betriebssystem geschaffen worden – GNU/Linux.

Linux, dessen Entwicklung nach wie vor von Linus Torvalds koordiniert wird, steht seit der Version 0.99.10 unter der GNU General Public Licence und ist als GNU/Linux heute führend im Supercomputing-Bereich mit einem Marktanteil von über 85% [vgl. *TOP500*]. Bemerkenswert ist auch der rasante Anstieg der Verbreitung in diesem Bereich von durchschnittlich ca. 10% pro Jahr seit dem Jahr 2000. Im Desktop-Bereich noch eine Randerscheinung verzeichnet das freie Betriebssystem jedoch auch hier das proportional größte Wachstum unter den Betriebssystemfamilien.

3.5 General Public Licence – Copyleft

Die GNU/GPL wurde im Januar 1989 von Richard Stallman verfasst und liegt mittlerweile in dritter Version vor. Unter ständigem Beschuss der Vertreter des „Copyrights“ musste die Lizenz, welche eine der wichtigsten freien Lizenzen der „free software“ Bewegung darstellt, angepasst werden, um den Zweck, die vier von der FSF postulierten Grundfreiheiten zu garantieren, weiterhin erfüllen zu können. Die GPL garantiert Freiheiten und vor allem deren Fortbestand, was nicht ohne Restriktionen geschehen kann. So unterliegt eventuelle veränderte Software automatisch wieder der GPL, ist also an diese permanent gebunden und darf ebenfalls frei kopiert, studiert und verändert werden. Im Falle freier Software, lizenziert unter der GPL, könnte man also zurecht von Gemeinschaftseigentum sprechen.

3.6 Offene Standards

Um unabhängig von spezifischen Implementierungen eine zuverlässige Verbindung zwischen einzelnen, durchaus sehr unterschiedlichen Komponenten eines Systems zu erlauben, bedarf es einer neutralen Schnittstellenbeschreibung – eines offenen Standards.

> *„Eine unverzichtbare Grundlage für diese flexible Zusammenarbeit bilden standardisierte Schnittstellen, [...]“ [STOSS, S.36]*

Das IDABC (Interoperable Delivery of European eGovernment Services to public Administrations, Business and Citizens) definiert einen „offenen“ Standard wie folgt:

- *The standard is adopted and will be maintained by a not-for-profit organisation, and its ongoing development occurs on the basis of an open decision-making procedure available to all interested parties (consensus or majority decision etc.).*
- *The standard has been published and the standard specification document is available either freely or at a nominal charge. It must be permissible to all to copy, distribute and use it for no fee or at a nominal fee.*
- *The intellectual property - i.e. Patents possibly present - of (parts of) the standard is made irrevocably available on a royaltyfree basis.*
- *There are no constraints on the re-use of the standard.*

[IDABC, S.9]

In einer Studie der MA14 des Magistrats Wien hält man sich weitgehend an folgende Definition:

- *Die Anwendung des Standards ist entweder kostenlos oder zu geringen Kosten möglich*
- *Alle diesen Standard bildenden Spezifikationen wurden veröffentlicht*
- *Der Standard wurde in einer offenen Entscheidungsfindung erarbeitet*
- *Die Spezifikationen und die damit verbundenen Urheberrechte wurden einer Non-Profit Organisation zur Betreuung übergeben*
- *Die Weiterverwendung des Standards ist nicht eingeschränkt*

[STOSS, S.36]

Bruce Perens' Definition zugrunde ist ein offener Standard weit mehr als nur eine Spezifikation. Die Prinzipien hinter einem Standard bzw. die Praxis im Umgang mit diesem sind es, die einen Standard offen machen.

- *Availability: Open standards are available for all to read and implement.*
- *Maximize end-user choice: Open standards create a fair, competitive market for implementations of the standard. They do not lock the customer into a particular vendor or group.*
- *No royalty: Open standards are free for all to implement, with no royalty or fee. Certification of compliance by the standards organization may involve a fee.*
- *No discrimination: Open standards and the organizations that administer them do not favour one implementor over another for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low- and zero-cost implementations to be validated, but may also provide enhanced certification services.*
- *Extension or subset: Implementations of open standards may be extended, or offered in subset form. However, certification organizations may decline to certify subset implementations, and may place requirements upon extensions (see Predatory Practices).*
- *Predatory practices: Open standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute and sell software that is compatible with the extensions. An open standard may not otherwise prohibit extensions.*

[OSPP, Principles]

Für Perens hat ein offener Standard also den Prinzipien der maximalen Verfügbarkeit und Freiheit für den Benutzer zu folgen und soll des Weiteren auch mit keiner Lizenzgebühr oder ähnlichen Einschränkungen versehen sein. Seine Ansichten erfahren weltweite Akzeptanz und hohen Anklang in den Free Open Sources Software Communities. Ein offener Standard ist somit ein Regelwerk aus Rahmenbedingungen und Spezifikationen, die von einer unabhängigen Organisation bestätigt wurden bzw. generell akzeptiert und weit verbreitet von der Industrie genutzt werden. Zudem sollte die Verwendung und Verbreitung in keiner Weise eingeschränkt sein. Um als Standard funktionieren zu können, muss in der Praxis jeder Standard offen sein, womit das Attribut „offen“ redundant erscheint.

Obwohl unzählige Definitionen von Standards existieren, kann man erkennen, dass sie alle die folgenden Charakteristika aufweisen:

- ✔ Ein offener Standard wird in einem offenen Prozess entwickelt, der es jedem ermöglicht, am Entstehungsprozess teilzunehmen.
- ✔ Die Spezifikationen sind für jeden zugänglich, einfach zu lesen und anzuwenden.
- ✔ Es existiert keine wie auch immer geartete Kontrolle einer bestimmten Gruppe oder Organisation über diesen Standard.

Bekannte Beispiele offener Standards sind:

- HTTP : Hypertext Transfer Protocol (Text-Übertragungsprotokoll)
- HTML: Hypertext Markup Language (Auszeichnungssprache)
- ODT: Open Document Text (Dokumententextformat)
- SIP: Session Initiation Protocol (Netzprotokoll zur Steuerung von Sprachkommunikation)
- XMPP: Extensible Messaging and Presence Protocol (Universelles Nachrichtenprotokoll)

Standards sind für die moderne Gesellschaft von hoher Notwendigkeit und enormem Nutzen. Sie stellen die adäquate Qualität und Interoperabilität von Produkten und Diensten verschiedenster Hersteller sicher. Idealerweise gewährleisten sie so auf ökonomische Weise auch eine Steigerung der Qualität, der Sicherheit und Verlässlichkeit sowie der Effizienz und Interoperabilität.

3.7 Communities

Der ohne Übertreibung wichtigste Grundpfeiler eines Open Source Projektes ist die dahinter stehende Community. Im Rahmen von freier Software definiert sich eine Community (engl. Gemeinschaft, Gemeinwesen) als organisierte, freie Interessengemeinschaft von Entwicklern und reinen Benutzern, die nicht nur Ideen und Methoden, sondern auch Programmcode austauschen und keinen Zwang auf ihre Mitglieder ausüben. In ihrem Text „Open Source – Konzept, Communities und Institutionen“ fassen Nüttgens und Tesei sowohl die Gemeinsamkeiten aller Open Source Communities als auch deren Unterschiede in folgender Tabelle zusammen.

Gemeinsamkeiten	Unterschiede
• Internationale Zusammensetzung der Mitglieder • Räumlich, meist global verteilte Mitglieder • Internet als Kommunikationskanal • Freiwilliges und unbezahltes Engagement der meisten Mitglieder • Sehr wenig bezahlte Entwickler • In bestimmten Phasen extrem hohe Entwicklungsgeschwindigkeit und -dynamik • Hohe Ansprüche an Qualität, Flexibilität und Stabilität der Software • Finanzielle Unterstützung durch Spenden	• Phase der Entstehung • Aktuelle Anzahl der aktiven Mitglieder • Art und Anzahl der Benutzer • Art und Anzahl der unterstützten Plattformen • Lizenzpolitik • Kooperationspolitik mit kommerziellen Interessengruppen

Abb. 3: Vergleichstabelle OS Communities [KCI, S. 11]

Als Gemeinschaftsprojekt definiert lebt ein solches Projekt nicht nur von den Programmierern, die sich zusammenschließen, sondern auch von den Testern und den einfachen Benutzern, die häufig auch als Supporter in Foren oder Mailinglisten zur Verfügung stehen.

Für die Koordinatoren von freien Softwareprojekten gilt es als oberstes Gebot, Beta-Tester und Co-Entwickler in alle Prozesse mit einzubinden und sie jederzeit auch bei wichtigen Entscheidungen zu integrieren, um bei ihnen das Gefühl zu steigern, ein wichtiger Teil des Projektes zu sein. Für den Projektleiter eines FOSS-Projektes sind also nicht nur Programmierkenntnisse, sondern vor allem auch Soft Skills notwendig. Den Zusammenhalt einer Interessengemeinschaft sichern darüber hinaus gemeinsame Visionen und Ziele, vereinbarte Beziehungen, gerechte Aufteilung und selbstverständlich engagierte Teilnehmer.

Zu Zeiten, in denen die ersten Computer entstanden, war eine solche Zusammenarbeit eine Selbstverständlichkeit. Sämtlicher Code wurde ausgetauscht und es war jedem mit dem nötigen Wissen möglich, diesen den eigenen Bedürfnissen entsprechend zu erweitern und zu verbessern. Erst in den frühen 80er-Jahren begannen die Leute, ihren Code zu verschließen und dessen Veränderung zu verbieten, selbst wenn es notwendig war, diesen aus funktionalen oder sicherheitsrelevanten Gründen zu bearbeiten. Wichtigster Pionier des proprietären Softwaremodells war zu dieser Zeit Microsoft.

> *„And that put me into a moral dilemma, you see, because to get one of the modern computers of the day, wich was the early 80s you would have to get a proprietary operating system, the developers of those systems didn't share wich other people, instead they tried to control the users, dominate the users, restrict them, saying if to get this system you have to sign a promise you wont share with any body else, and to me that was essentially a promise to be a bad person, to betray the rest of the world, cut myself of from society, from cooperating community. It kept us from doing useful things.“ [REVOS, Min. 09.55]*

Richard Stallman fühlte sich durch die Veränderungen, insbesondere durch die Wegnahme der Freiheiten bei der Verwendung von Software in seinem Handeln eingeschränkt und beschloss daraufhin, ein eigenes freies Betriebssystem zu entwickeln. Dies führte Mitte der 80er-Jahre zur Gründung der Free Software Foundation und der steten Entwicklung von GNU/Linux unter der GNU General Public License, welche die Freiheiten der Software sichern sollte.

Heute sind Communities längst nicht mehr auf Gruppen von Programmieren beschränkt. „*http://brainstorm.ubuntu.com*“, „*http://bugs.kde.org*“ etc. sind als Vertreter von Web2.0 Systemen in erster Linie an die einfachen Konsumenten gerichtet. Bewertungssysteme, Wunschlisten, Fehlerlisten, usw. schaffen für alle Benutzer Möglichkeiten der Mitarbeit und den einfachen Zugang zur Community. Nicht zuletzt binden eine kaum überschaubare Anzahl von Community-Foren auch die wenig versierten Nutzer freier Software in die Open Source Welt ein und erlauben es, Wissen auszutauschen und einander zu helfen.

3.8 Open Source Business

Bill Gates war zur Entstehung der ersten massentauglichen Computer einer der bekanntesten Verfechter von „Closed Source Software“. In seinem „offenen Brief an die Hobbyisten“, beschuldigte der ehemalige Microsoft CEO die Entwickler freier Software des Diebstahls, brachte seinen generellen Unmut diesen gegenüber zum Ausdruck und warf ihnen vor, die Entstehung guter Software zu verhindern. [vgl. *BILL*]

„Who can afford to do professional work for nothing?“ [BILL]

Wie man heute leicht feststellen kann, machte Bill Gates bei seiner Einschätzung den Fehler, dass er davon ausging, die Entwickler würden die ganze Arbeit der Programmierung bis hin zur Fehlersuche alleine vollbringen müssen und, da sie den Code der Software unentgeltlich weitergaben, dass sie auch keinerlei Einnahmequellen haben würden. Er ahnte wohl nicht, dass eben diese Leute sich neue Wege ausdachten, um mit Software Geld zu verdienen, ohne den Anwendern sämtliche Rechte an dieser zu nehmen, die Arbeit größtenteils in ihrer Freizeit erledigten, unzählige Freiwillige in den Entwicklungsprozess einbinden und unter anderem auch für die Entwicklung freier Software bezahlt werden würden.

„[…]die Autoren Freier Software (berechnen) in der Praxis ihren Kunden selten die Entwicklung der Anwendung, sondern schreiben Rechnungen für die Zusammenstellung, den Versand, die Anpassung und die Kundenbetreuung. Aber auch wenn der Entwickler seinen Kunden für das Schreiben der Software zur Kasse bittet, um damit auch die Qualität der Anwendung garantieren zu können ist das aus unserer Sicht unproblematisch.“ [FSPIR]

Products:	**Packaging**	Facilitating and distributing open source products for easy use.
	Proprietizing	Adding proprietary solutions to an open source product and selling licenses for the new product.
	Spin-off	Selling complementary proprietary software products surrounding a major open source project.
	Black-box	Bundling and integrating pieces of open source software products in a hardware solution.
Services:	**Education and training**	Education based on an area of expertise within open source software.
	Consultancy	Consultancy work based on an area of expertise within open source software.
	Support	Support based on an area of expertise within open source software.

Abb. 4: Business-Modelle, [MFOSC, S.16]

Wie aus der Übersichtstabelle ersichtlich, lassen sich daraus noch weitere Geschäftsmodelle ableiten, die durchaus profitabel eingesetzt werden können. Free Open Source Software und geregeltes Einkommen stehen in keiner Weise im Widerspruch. Auch wenn ein beachtlicher Teil der Software von Freiwilligen und Studenten an Universitäten erstellt wird, so wird die Anzahl derer, die mit offenem Code unter freien Lizenzen ihr Geld verdienen, immer größer. Allen voran IBM, Nokia, Novell und Sun Microsystems sowie eine wachsende Anzahl diverser Klein- und Mittelunternehmen. Insbesondere letztere können sowohl den Studien der Stadt München als auch jener der Stadt Wien zufolge enorm vom verstärkten Einsatz freier Software in öffentlichen Einrichtungen profitieren.

„Durch die Erteilung von lokalen Entwicklungsaufträgen können auch erhebliche Impulse für die lokale Wirtschaft geleistet werden. Während die Zahlung von Lizenzen kaum positive regionalwirtschaftliche Auswirkungen hat, kann durch lokal entwickelte Software große Dynamik erreicht werden.[...] Die bewusste Nutzung der Möglichkeiten des frei zugänglichen und verfügbaren Programmcodes eröffnet zahlreiche Möglichkeiten zur Kooperation und zur Stärkung der lokalen Wirtschaftsbetriebe. Damit können die schnellere Verfügbarkeit von Programmen, die Ersparnis von Entwicklungskosten und lokale Wirtschaftsimpulse erzielt werden.“ [STOSS, S.24]

Freie Software in öffentlichen Einrichtungen führt also indirekt zur Förderung regionaler Kleinunternehmen, da eventuelle Anpassungen und Erweiterungen jener Software ebenso wie der Support für diese an viele verschiedene Unternehmen in Form ordentlicher Ausschreibungsverfahren vergeben werden kann. Dies ist eine logische Folge des Fehlens einer Quasi-Monopolstellung, wie sie im Fall von proprietärer Anwendungssoftware zumeist anzutreffen ist. Dort wird Support zu einem Monopol. Es gibt nur eine Firma, die den Quellcode der Software besitzt und somit das Know-how, um guten Support geben und im Falle eines Fehlers diesen zu beheben. Die Firma CYGNUS war eine der ersten Firmen, die sich vollkommen auf FOSS spezialisierte und nach wie vor Support für solche Software anbietet. Von Beginn an gab es in der Free Software Bewegung Platz für geschäftliche Interessen.

Wie bereits erwähnt ist es eine der bemerkenswertesten Eigenschaften dieses Konzepts, dass ein großer freier Markt für jede Art von Supportleistung und andere Services existiert. Setzt man auf freie Software, so stehen eine hohe Anzahl an Firmen zur Auswahl, die für Supportleistungen in Frage kommen würden. Zudem sehen sich jene Firmen in der Situation, dass sie herausragenden Support zu leisten haben, da sie sonst ihre Kunden an andere verlieren würden, was zu einer regen Konkurrenz um Supportaufträge führt und somit billigere und bessere Leistungen zur Folge haben kann.

Sun Microsystems' CEO John Gage postuliert im Interview mit Wolfgang Stieler noch weitere Gründe, mit offenem Code zu arbeiten:

> *TR: Wenn Sie also Ihre Software verschenken, wie machen Sie dann Ihr Geld?*
> *Gage: Oh, das ist im wesentlichen dasselbe wie bei Java. Die meiste Java-Software wird von anderen entwickelt, von Nokia oder IBM oder Siemens. Es ist nicht so, dass wir unser Geld mit Software machen. Es geht mehr um die Programmierumgebung. Wenn das Niveau steigt, wird ein gewisser Prozentsatz der Maschinen, die gekauft werden, aus Sun-Computern bestehen.*
> *Der zweite Pfad - ich kann Ihnen allerdings jetzt keine Zahlen nennen - ist die Offenlegung unserer Hardware vor sechs Monaten. Als Ergebnis haben in diesem Herbst 1000 chinesische Universitäten das Design unserer Multi-Core-Prozessoren in ihre Vorlesungen mit aufgenommen. Sie müssen uns nichts dafür bezahlen. Und sie können das nehmen und verändern.*
> *Wenn Ihnen nicht gefällt, wie Sun in dem neuen Niagra-Chip Fließkommazahlen verarbeitet, dann machen Sie es einfach selbst. Und Sie können einfach einen Dienstleister beauftragen, das dann zu fertigen - Sie brauchen heutzutage keine eigenen Fabs mehr. Sie können spezifische Verbesserungen für spezifische Anwendungen bauen und brauchen uns kein Geld dafür zu bezahlen. Aber das verbreitet unsere Art des Multi-Threadings schneller - also nützt es uns. [GAGE, S.2]*

John Gage postuliert damit zusammengefasst einen Vorteil für jede Firma, die es versteht, das Open Source Modell mit Weitsicht für sich einzusetzen. Auch andere Firmen wie Redhat und Novell setzen auf die Mithilfe der Communities, wenn sie eigene Änderungen am System an diese zurückgeben und selbst von der raschen Weiterentwicklung ihrer Community-Distributionen „Fedora“ und „openSUSE“ profitieren.

4 FOSS im Schulbetrieb

Das vorliegende Kapitel zeigt die Hürden, die sowohl bei freier als auch bei proprietärer Software im Schulbereich überwunden werden müssen und beschreibt, auf welche Weise sich eine Umstellung auf freie Software für Schulen vorteilhaft gestalten kann. Die Möglichkeiten für Verwaltung und Unterricht werden anhand praktischer Beispiele näher ausgeführt.
Wie die Untersuchung dieses Themas ergeben hat, scheinen pädagogische Aspekte bei der Wahl des Betriebssystems, einer der grundlegenden Entscheidungen beim Einsatz von Computern in der Schule, kaum vertreten zu sein. Bedienbarkeit, Pflegeaufwand, Anwendungssoftware und Kosten beeinflussen diese in erster Linie. Ethische und politische Überlegungen können in seltensten Fällen bei der Entscheidungsfindung mit einbezogen werden, spielen aber dort, wo es die Umgebungsvariablen erlauben und sich die Wahl der Software auf diese Werte zuspitzt, oft letztlich die entscheidende Rolle.

4.1 „Lock-in“ Bindung an einzelne Hersteller

Das Problem einer kontinuierlichen Bindung an einzelne Hersteller kann durch freie Software als Alternative und Weg aus dem Vendor Lock-in überwunden werden.

Als Lock-in-Effekt (engl. *to lock in: einsperren*) bezeichnet man in den Wirtschaftswissenschaften anfallende Kosten, die jede Änderung der aktuellen Situation, beispielsweise die Verwendung einer neuen Bürosoftware eines anderen Anbieters, unwirtschaftlich machen würden und zuweilen zu problematischem, verlustbehaftetem Verhalten zwingen, um noch größere Verluste abzuwenden. So versuchen große Unternehmen so weit wie möglich den eigenen Lock-in zu vermeiden und andererseits durch finanzielle und zeitliche Investitionen seitens der Kunden in Hardware und Support ein sogenanntes „Vendor Lock-in“ zu schaffen und dadurch jene Kunden zu binden. Dies führt zum Vorzug eines bestimmten Verkäufers bis zur Bildung von Quasi-Monopolen wie beispielsweise jene von Microsoft, im Bereich der Betriebssysteme und der Office-Software.

“The Windows API is so broad, so deep, and so functional that most ISVs would be crazy not to use it. And it is so deeply embedded in the source code of many Windows apps that there is a huge switching cost to using a different operating system instead. [...] It is this switching cost that has given customers the patience to stick with Windows through all our mistakes, our buggy drivers, our high TCO, our lack of a sexy vision at times, and many other difficulties. [...] Customers constantly evaluate other desktop platforms, [but] it would be so much work to move over that they hope we just improve Windows rather than force them to move. In short, without this exclusive franchise called the Windows API, we would have been dead a long time ago.”
[MCASE, S.126 ff]

Microsofts Anwendungssoftware schafft Abhängigkeit in allen Bereichen, unter anderem durch die Verwendung und Verbreitung von proprietären Datenformaten. So verwenden sowohl Outlook als auch Word Dateiformate und Schnittstellen, die es mangels offenen Quellcodes und gesetzlichen Restriktionen durch restriktive Lizenzen legal unmöglich machen, von anderen Programmen als jenen Microsofts gelesen zu werden. Für die Anwender entsteht dadurch eine auf den ersten Blick unauflösbare Bindung an das Produkt und es scheint, als wäre man gut beraten, die Software und somit auch den Hersteller beizubehalten, selbst wenn das Produkt unzulänglich ist und die individuellen Bedürfnisse nur teilweise abzudecken vermag.

Was in diesem Bereich durch freie Software erreicht werden kann, veranschaulicht sehr gut das Beispiel des freien Webbrowser „Firefox“, welcher es binnen kurzer Zeit geschafft hat, zu einer ernst zunehmenden Größe und Konkurrenz heranzuwachsen. Der Lock-in Effekt des Konkurrenzproduktes „Internet Explorer“ bestand in diesem Fall ausschließlich durch die von Microsoft geschaffenen Erweiterungen und Besonderheiten von Internetstandards wie CSS und HTML und der umstrittenen Eigenentwicklung „active-x“, die im Laufe der Zeit von Webdesignern angenommen und eingesetzt wurden, auf welche ich in dieser Arbeit jedoch nicht näher eingehen werde. „Designed for Internet Explorer 6 (IE)“ war bis vor kurzem noch ein weit verbreiteter Satz in der Welt des WWW.

Webdesigner aller Länder hatten sich auf den bereits auf jeder Maschine vorinstallierten IE eingearbeitet und programmierten nach dessen Richtlinien und nicht nach jenen des World Wide Web Consortium (W3C). Konkurrenzprodukte wurden damit in eine Ecke gedrängt, gebrandmarkt durch die allgemeine Ansicht der Unbrauchbarkeit wegen mangelhafter Webseitendarstellung, und blieben bzw. wurden Nischenprodukte. Microsoft prägte zu dieser Zeit die Bezeichnung **e**mbrace, extend and extinguish (EEE) für seine Unternehmensstrategie, welche als Herangehensweise an neue Produktsektoren beschrieben werden kann, in welchen man sich, ausgehend von verbreiteten Standards, etabliert, in dem man diese durch proprietäre Technologien erweitert. In der Folge werden diese flächendeckend verwendet, um die Konkurrenz zu benachteiligen.

Als Microsoft sich jedoch auf den Lorbeeren des in die Jahre gekommenen IE6 auszuruhen begann und die Kunden dies scheinbar akzeptierten, stieg Firefox wie Phönix aus der Asche und begeisterte in einem regelrechten Hype nicht nur die Communities, sondern bald auch alle anderen Benutzer des Internets. Die kostenfreie Verteilung über das Internet, eine starke Community und die aktive Einbindung eines jeden, der ein wenig programmieren konnte, brachten es fertig, dass der Webbrowser Firefox inzwischen bereits in der dritten Version verfügbar ist und einen Marktanteil von über 30%, Tendenz steigend, in Europa und über 45% in Finnland [vgl. *XMON*] für sich in Anspruch nimmt. Darüber hinaus veranlasste „David“ auch „Goliath“ (Microsoft) dazu, nach über fünf Jahren eine neue Version seines hauseigenen Browsers „Internet Explorer“ herauszugeben, um die Marktanteile nicht vollständig zu verlieren, sowie sich verstärkt an internationale Standards zu halten, denen man sich im Webdesign dank Firefox inzwischen wieder genähert hat. Zu spüren bekommen hat die Effekte des Vendor Lock-in auch die Stadt München im Jahr 2003, als Microsoft beschloss, den Support für das sich im Einsatz befindliche Windows-NT einzustellen. Der Wechsel auf ein neues Microsoft Betriebssystem, verbunden mit den dadurch notwendigen Hardwareupgrades und auch der Einführung des „Active Directory“ Systems, welches einen enormen Lock-in einschließlich hoher Folgekosten bedeutet hätte, stand der Einführung eines völlig neuen, unabhängigen Systems aus freier Software gegenüber.

So entschied sich die „Stadt München“ nach eingehenden Studien zu den Kosten eines Umstiegs auf freie Systeme statt zu notwendigen Neuerungen innerhalb des bestehenden proprietären Systems, für GNU/Linux und darauf lauffähige, freie Software in der Verwaltung, trotz des höheren Kostenaufwandes, den dieser zu Beginn bedeuten würde. Die Verantwortlichen des Projektes stellten die damit gewonnene Freiheit als Ursache für ihre Entscheidung in den Vordergrund.

> *„While the proprietary solution was deemed to be slightly more cost-effective over the full period, the strategic advantage of being free to take its own IT decisions led the city council to decide in favour of the migration to GNU/Linux.“ [OSOR, S.1]*

Was für eine Stadtverwaltung gilt, sollte vor allem im Bildungsbereich nicht vernachlässigt werden. Wirft man den Blick in die Richtung von Bildungseinrichtungen, zeigt sich jedoch ein anderes Bild. An Universitäten und Schulen ist es heute Usus, die Abgaben von Projektarbeiten, Präsentationen und andere Arbeiten in digitaler Form zu tätigen. Zum Zuge kommen hier vorzugsweise gängige proprietäre Lösungen. Das notwendige „Know-how“, um diese zu bedienen sowie der Besitz der Software wird all zu oft vorausgesetzt. So finden sich Schüler in der zweifelhaften Lage, Microsoft „Word“ bedienen zu müssen und „Powerpoint“ Präsentationen vorzubereiten. Architekturstudenten arbeiten mit Profisoftware wie Archicad und 3DMax, die einen realen Kostenaufwand im vierstelligen Euro Bereich bedeutet, und im Informatikunterricht wird Photoshop als Grafikbearbeitungsprogramm gelehrt.

In der Studie der Europäischen Kommission „Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU“ heißt es zu diesem Thema:

> *Education: avoid lifetime vendor lock-in for students*
> *The reason it seems desirable to promote the use of FLOSS in education (ICT education and more generally all educational activities that have a bearing on the cultural relationship with information technology) is threefold:*
>
> *1. It is obviously likely to have a strong impact on the future usage of FLOSS products and the build-up of the related skills.*
>
> *2. It builds up essential ICT skills rather than the knowledge of specific applications from specific vendors (leading to the current locked-in-for-life situation, where vendor lock-in applies not only to organisations but to individuals who have typically not chosen their software but been provided it for free by schools).*
>
> *3. It is likely to install an attitude towards information technology that favours the ability to create and actively participate rather than just consume – i.e. The scenarios under which FLOSS is most likely to deliver a strong positive economic and societal impact, by encouraging collaborative prosumer usage and a reflexive attitude on usage and the technology that supports it. [FLOSSI, S.216]*

Hier ist die Rede vom Lock-in des Individuums als solches, allein durch die Lehre in der Schule, welche die Verwendung proprietärer an Stelle freier Lösungen setzt. Die spezifischen Softwarekenntnisse, die vermittelt werden, bedeuten für viele eine lebenslange Abhängigkeit vom Gelernten und somit vom entsprechenden Hersteller der Software.

„[...] das ist natürlich überhaupt ganz tendenziell ein Problemfeld, weil du ... wenn du dir überlegst, dass du z.B. Schulungen für proprietäre Software machst ... tust du da nicht quer subventionieren? Wir haben z.B ein Hilfsmittel in dieser EDV-Einführung „Einführung in Mathematica“. Indirekt sponserst du damit natürlich Wolfram aus staatlichen Geldern, weil du die Leute daran gewöhnst, es zu benutzen und es später wahrscheinlich das Erste sein wird, was sie in die Hand nehmen.“ [Florian Wisser, Interview 2008]

Was ist nun der Unterschied zwischen der Abhängigkeit von freier Software und jener, die im Besitz einer Firma ist und deren Gesetzen gehorcht? Wie es aus der Studie hervorgeht, schafft die Nutzung freier Software ein Mehr an Freiheit und Mündigkeit. Es kann gelingen, die Schulpflichtigen aus ihrer Position als reine Konsumenten zu holen und sie zur aktiven Teilnahme sowohl an der Softwareauswahl als auch an der Entwicklung zu bewegen. Überdies unterscheidet sich die Art der Abhängigkeit durch gelernte Fertigkeiten, die an eine spezifische Software binden, drastisch, vergleicht man hier proprietäre mit freien Softwarelösungen. Letztere zeigt sich schließlich in einem wesentlich flexibleren Rahmen, ohne spezifischem Anbieter und Lizenzbestimmungen sowie anfallenden Kosten.

Schenkt man den Prognosen der Bildungsstrategen Glauben, so wird die informationstechnische Bildung eine der wichtigsten Schlüsselqualifikationen der Zukunft sein. Schon jetzt lässt diese sich nicht mehr auf die Bedienung einzelner Programme reduzieren. Dennoch macht sich der Eindruck breit, die Schüler würden in der Berufswelt nicht Fuß fassen können, sofern sie nicht bezeugen können, die Bedienung von Microsoft Office zu beherrschen. Ständige Innovation und Release-Druck beherrschen den Softwaremarkt und garantieren somit, dass jedes spezifische Wissen über die Bedienung einer Anwendungssoftware in dem Moment obsolet ist, sobald die Schüler in das Berufsleben eintreten. Aus diesem Grund muss das Erlernen von Konzepten anstelle von Produkten vorrangig sein, was auch in einem späteren Abschnitt zum „Einsatz von freier Software in der Schule“ dargelegt wird.

4.2 Finanzielle Konsequenzen von freien Softwarelösungen

Eine ganzheitliche Planung ist die Grundvoraussetzung für eine erfolgreiche Migration, die sowohl eine finanzielle Besserstellung als auch einen geringeren Wartungsaufwand bedeuten kann.

Während bestimmte freie Software möglicherweise bessere Ergebnisse im Unterricht erzielen könnte als proprietäre Software, gibt es dem ersten Anschein nach keinen vernünftigen Grund, warum dies generell so sein sollte. Daher liegt das Hauptaugenmerk bei den ethischen und politischen sowie bei den finanziellen Aspekten, was eine strategische Herangehensweise an eine finanzielle Planung, in welcher die Kosteneinsparungen in bestmöglicher Weise dem Unterricht und somit dem Lernprozess zugute kommen, wichtig erscheinen lässt.

> *„However, in general, anything that reduces the cost of some part of the mechanics behind the teaching and learning activities potentially helps learning by freeing resources that can be used in other ways. Hence cost saving, as distinct from cost cutting, through the introduction of OSS, could lead indirectly to better educational attainment.“ [BECTA, S.15]*

Der Informatik-Etat einer Schule bzw. des Ministeriums wird also nicht durch teure Programmlizenzen belastet, wodurch mehr Geld für die Hardware zur Verfügung steht. Zudem müssten auch keine kostenpflichtigen Probelizenzen angeschafft werden, was dazu führt, dass ohne Risiko getestet werden kann, welche Software für den Unterricht am besten geeignet ist, was in weiterer Folge Raum für Innovationsmöglichkeiten schafft.

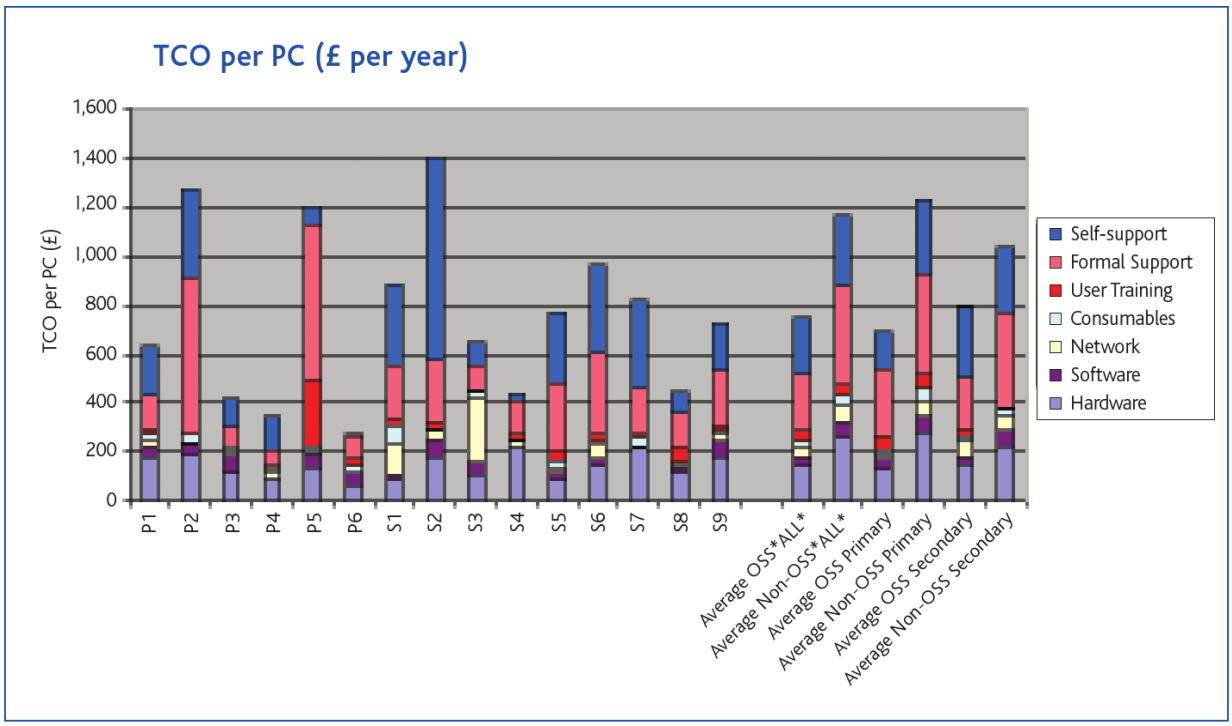

Abb. 5: Jährliche TCO pro PC [BECTA, S.10]

Die der britischen Studie der BECTA entnommene Grafik, welche sowohl die Kosten für Software als auch die damit verbundenen Nebenkosten aufschlüsselt, zeigt hier eine durchschnittliche Kostenersparnis von ca. 400 britischen Pfund (umgerechnet etwa €580.- Euro) pro PC pro Jahr für jene Schulen, die freie Software einsetzen. Diese Ergebnisse sind jedoch auf Grund folgender wesentlicher Aspekte mit Vorsicht zu betrachten: Zunächst einmal variieren die Kosten für die einzelnen Teilbereiche der Schulen untereinander enorm. So würden weitere Mittelschulen mit einer Kostenaufteilung ähnlich der Schule „S2“ anstelle von „S4“ und „S8“ diese Statistik von Grund auf verändern.

Auch wird in vielen Schulen dieser Studie „Star Office“ zum freien Softwarepaket gezählt, welches nach bereits angeführter Definition keine vollständig freie Software und unter anderem auch kostenpflichtig ist. Zudem geht aus den Daten hervor, dass die Vergleichsgruppe der Schulen, die keine freie Software einsetzen - was evident ist in Anbetracht der Vielfalt an Variablen in komplexen Systemen wie diesen - in ihrer inneren Zusammensetzung von jener der reinen „FOSS“-Schulen abweicht.

Schließlich geht aus der Beschreibung der Studie hervor, dass keine der Schulen tatsächlich 0% bzw. 100% freie Software einsetzt und wir in diesem Falle eine Auswahl an Schulen betrachten, die höchstens eine Annäherung an die Wirklichkeit darstellen kann. Dennoch erlaubt uns die angeführte Verteilung der Kosten der einzelnen Schulen einen Überblick und zeigt ein unmissverständliches Ergebnis, welches auch von anderen Studien, wie der Studie OSS der MA14 Wien [vgl. *STOSS*], bestätigt wird. Dies bezieht sich jedoch auf die laufenden Kosten, nicht auf den Migrationsaufwand.

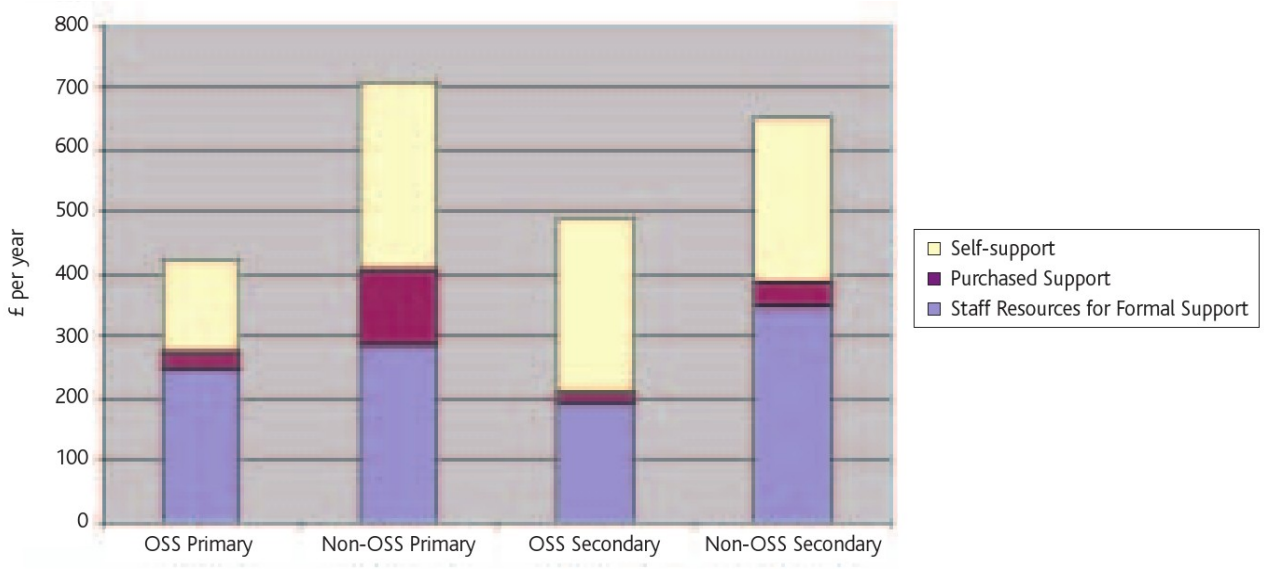

Abb. 6: Support-Kosten [BECTA, S.12]

Wie man Abbildung 6 entnehmen kann, hat die Untersuchung vor allem bezüglich der Supportkosten einen klaren finanziellen Vorteil für Schulen ergeben, die freie Software einsetzen. Zudem ließ sich auch feststellen, dass Schulen, die zu annähernd hundert Prozent auf freie Software setzen, niedrigere Kosten zu tragen haben als jene, die auf ein Mischsystem bauen. Dies begründet sich zum einen durch den höheren Wartungsaufwand, den zwei verschiedene Systeme zur Folge haben, zum anderen aber auch in technischen Vorteilen, die Linuxsysteme für die Verwaltung mittelgroßer Schulnetze zu bieten haben, wie folgende drei aus dem Leben gegriffene Zitate zu veranschaulichen vermögen.

„Was bei uns dafür gesprochen hat, ist erstens natürlich der Preis, zweitens ein wichtiger Punkt, finde ich auch, ist die Administrierbarkeit; also jetzt haben wird das vom ZID drauf gespielt, Kubuntu, mit $HOME Verzeichnissen am NFS Server. Es ist zum Beispiel so, wenn ein Linux Computer eingeht, dann stellst du ihn weg, stellst einen neuen PC her, sagst ihm, welcher Client er werden soll, drehst ihn auf, er installiert sich, der User loggt sich ein und es hat sich für ihn nichts geändert. Probiere das einmal konsistent in einem Windowsumfeld zu machen, das ist NICHT leicht. [...] also in Wahrheit: von der Administration ist Linux viel viel angenehmer." [Florian Wisser, Interview 2008]

„Ein System braucht immer Wartungsaufwand, ganz egal, ob das ein Windowssystem ist oder ein Linuxsystem, wartungsfrei wird es nicht gehen; wir haben jetzt die Möglichkeit, es wartungsarm zu machen mit den ganzen Linux USB-Stick Varianten und Überlegungen, was mit einem Windowssystem jetzt nicht möglich wäre." [Rene Schwarzinger, Interview 2008]

„Für einen Umstieg steht in erster Linie die Administration der Software aus Sicht des Computeradministrators, weil ich sehe, wie sich der oft plagt, wenn er irgendwelche Sachen wiederherstellen muss bei Windowssystemen, weil das alles so ein starres System ist; man kommt kaum dazu einzelne Sachen wiederherzustellen." [Marco Delbello, Interview 2008]

Wie man aus den Antworten der Interviewpartner lesen kann, besteht eine klare Übereinstimmung der Auffassung, dass Linuxsysteme den Wartungsaufwand senken würden, was im Falle einer Migration in jedem Fall beachtet werden sollte.

Die Stadt München hat eine Migration bereits in Angriff genommen und für diese vorausgehende Studien in Auftrag gegeben, welche sich u.a. intensiv mit den Migrationskosten beschäftigten und hier verständlicherweise einen beachtlichen Mehraufwand im Falle eines Umstiegs gegenüber dem Beibehalten des bestehenden Systems bescheinigten. So heißt es in der Studie der Stadt München:

„The Microsoft solution would have made it necessary to introduce an Active Directory system, which would have meant a strong lock-in and would have caused significant follow-up costs. The total cost for the proprietary solution were calculated to be 35 million Euro, against 37 million Euro for GNU/Linux (both including all costs beyond the solution itself, such as personnel and training costs, over five years). While the proprietary solution was deemed to be slightly more cost-effective over the full period, the strategic advantage of being free to take its own IT decisions led the city council to decide in favour of the migration to GNU/Linux.“ [OSOR, S.1]

Zwei Jahre nach der Vorstudie zum Projekt „LiMux“, die eine Kostenspanne von 3 Millionen Euro zusätzlich für den Umstieg auf freie Software festlegte, kann man bereits die Früchte der Entscheidung zur anfänglich zwar kostenintensiveren, aber von Lock-in Effekten freien Variante bereits ernten und sieht dabei einer rosigen Zukunft entgegen. Die aufgewandten Kosten amortisieren sich nach Angaben der Stadt München über fünf Jahre allein durch die Einsparungen an Lizenzkosten. [vgl. *MUENCH*]

So lassen sich die Vorteile der freien Software für die Stadtverwaltung wie folgt zusammenfassen:

„Einsparungen und Förderung von KMU: Durch den Einsatz von Open Source bezahlt die Stadt nicht mehr vorrangig für die Beschaffung der Software selbst, sondern nur für die notwendige Anpassungen an die besonderen Kundenwünsche. Diese Flexibilität können große Firmen und kommerzielle Programme oftmals nicht bieten, so dass freie Software und offene Standards es vor allem kleinen und mittleren Unternehmen aus der Region ermöglichen, sich umfassend am Wettbewerb zu beteiligen. So wurde im Rahmen des LiMux-Projekts bislang ein Drittel der externen Projektmittel an KMU vergeben.

Effizienzsteigerung: Durch eine konsequente Vereinheitlichung und Konsolidierung z. B. der städtischen Office-Vorlagen wird eine Verringerung von Formularen um ca. 30% erreicht. Geschäftsprozesse, vor allem im Office-Bereich, werden optimiert. Wenn für sehr viele Beschäftigte die Arbeit ein Stück einfacher wird, ergibt dies zusammengerechnet pro Jahr die Arbeitsleistung von ca. 80 Mitarbeitern, die künftig besser genutzt werden kann.

Offene Standards: Die Stadtverwaltung setzt künftig verstärkt auf offene Standards wie den von der ISO definierten Standard «Open Document Format» (ODF). Durch die offene Beschreibung des Formats können ODF-Dateien nicht nur von dem Programm eines Herstellers gelesen werden. Das bedeutet Unabhängigkeit von einem oder wenigen Herstellern. Mit offenen Standards können Monopolstellungen vermieden und der Wettbewerb gefördert werden.

Zukunftssicherheit: Die aktuelle Migration schafft die Grundlage dafür, dass zukünftige Anpassungen mit geringem Aufwand durchgeführt werden können. Einerseits durch Einsparung von Lizenzkosten in Höhe von über 3 Mio. Euro in den nächsten fünf Jahren (Wegfall von Ersatzbeschaffungen/Updates für Windows und MS-Office etc.), andererseits durch die Produkt- und Hersteller-Unabhängigkeit." [MUENCH]

4.3 Offene Schnittstellen und offene Standards

Offene Standards stehen für einen reibungslosen Austausch von Dateien, langfristige Sicherstellung der Lesbarkeit von Daten und Freiheit der Softwarewahl.

(Offene Standards?) „Grundsätzlich.. absolut wichtig!!" [Florian Wisser, Interview 2008]

Der freie Zugang zu Bildung und Information ist einer der Grundpfeiler unserer Gesellschaft. Techniken werden standardisiert und dokumentiert. Eines der Probleme, die ein proprietäres System mit sich bringt, bei dem auch das Wissen als Eigentum gehandhabt wird, zeigt sich sehr deutlich, betrachtet man die Schwierigkeiten, mit denen man heute bereits beim Aufbewahren und Erhalten von Informationen konfrontiert wird.

Für Behörden, Bibliotheken, aber auch Schulen sind offene Standards ein zunehmend wichtiges Thema. Für sie gibt es abgesehen von den enormen Datenmengen zwei Probleme: die Kurzlebigkeit physischer Medien, auf welchen die Daten gesichert werden und die Dateiformate, in denen Schriften, Videos und Bilder gespeichert werden. Langfristig führt für sie kein Weg an offenen, international standardisierten Formaten vorbei. Bei diesen liegt das Format bzw. die Spezifikation desselben offen und kann auch in hundert Jahren noch von Entwicklern nachgebaut werden.

An dieser Stelle tritt die freie Software-Szene auf die Bühne, die sich die Offenheit ihrer Systeme und die Einigung auf gemeinsame Standards auf die Fahne geschrieben hat.

> *„OSS und offene Standards unterstützen effizient einen auf die langfristigen Anforderungen ausgerichteten IT-Einsatz. Sie bieten durch die Art des Entwicklungsprozesses und ihre freien Lizenzbestimmungen eine signifikante Alternative zu herkömmlichen Angebotsformen.“*
> *[STOSS, S.21]*

Für öffentliche Einrichtungen, insbesondere deren Verwaltungen wird die Informationstechnologie immer mehr zur unverzichtbaren Grundlage der täglichen Arbeitstätigkeit. Sowohl die vermehrte Anzahl an Aufgaben, die die Hilfe von Prozess unterstützender Software bedingen, als auch die unaufhaltsam steigende Vernetzung innerhalb von Schulen, zwischen den einzelnen Bildungseinrichtungen sowie zur privaten Außenwelt der Lehrenden und Lernenden, ließen eine perfekt funktionierende Zusammenarbeit unterschiedlicher Systeme und Programme in jüngster Zeit zu einem wichtigen Kernthema heranwachsen. So heißt es in der Studie der MA 14, Wien:

> *„Offene Standards sind eine wesentliche Voraussetzung für die Interoperabilität in heterogenen Systemen. Nur wenn die Schnittstellen vollständig definiert und deren Festlegungen frei zugänglich sind, bleiben die Investitionen geschützt und die Entscheidungsfreiheit dauerhaft erhalten.“ [STOSS, S.35]*

Diese Erkenntnis ist auch für Lernende ein wichtiger, grundlegender Schritt. Programme arbeiten nicht deswegen problemlos zusammen, weil sie eine ähnliche grafische Benutzeroberfläche haben (MS Office Suite, Adobe Creative Suite, etc.) sondern, weil sie mit austauschbaren bzw. sich ergänzenden Dateiformaten arbeiten, die nicht, wie es bei proprietärer Software implizit vermittelt wird, an die Anwendungen selbst gekoppelt sein müssen. Es ist wichtig, schon in der Schule zu lernen, dass es produktunabhängige Dateiformate gibt sowie unterschiedlichste Anwendungssoftware, die mit diesen arbeiten kann. Nur so können die Schüler die Bedeutung von offenen Standards erfassen.

Ein für Schulen weniger relevanter, aber dennoch durchaus interessanter Aspekt von offenen Schnittstellen, der noch erwähnt werden sollte, ist, dass diese vor allen Dingen für Programmierer von großer Bedeutung sind.

> *„Wenn ich in der Lage, bin Sachen dazu zu programmieren bzw. verschiedene Programme miteinander zu verbinden, ist das ganz sicher von Vorteil, wenn diese Schnittstellen offen sind.“ [Marco Delbello, Interview 2008]*

Nicht offengelegte Schnittstellen bedeuten für Programmierer in der Regel „Reverse Engineering“, was bedeutet, dass diese durch Untersuchung von Zuständen, Strukturen und Verhaltensweisen des Produktes die einzelnen Konstruktionselemente extrahieren und aus dem fertigen Objekt wieder die „Blueprints“ zu erstellen versuchen. Ein Unterfangen, welches sich durch geschlossenen Code, Standards und Schnittstellen als sehr zeitaufwendig und teilweise auch unmöglich erweisen kann und somit die Entstehung und Evolution von Software behindert.

Aaron Bockover, Entwickler des freien Mediaplayers Banshee, schreibt dazu auf seinem Blog zum Banshee Release 1.4 mit Google Phone support:

> *„A final note on the G1 support: because the Android platform is open source, I was able to easily figure out optimal ways of implementing Android/G1 support. For instance, I was unsure what the maximum cover art size should be on the device, so I just read the source. It was a nice for once to not have to reverse engineer or guess!“ [ABOCK]*

4.4 Aspekte der Wiederverwertbarkeit freier Software

Weitergabe von Software und deren Verbesserungen fördert sowohl die Kooperation zwischen den Beteiligten einer Schule als auch zwischen den Schulen untereinander.

> *„Es kann nicht angehen, dass die Schüler darauf trainiert werden, proprietäre Software zu bedienen, die sich dann der Kompatibilität zu Willen virusartig zu Hause und im Freundeskreis verbreitet. Hier sind wir auch am Kernproblem der proprietären Software angelangt. Proprietäre Software funktioniert immer nur mit sich selber gut beziehungsweise mit Programmen, die vom Entwickler für gut befunden wurden. Diese softwaretechnische Intoleranz kann unmöglich von Staatswegen gefördert werden.“ [FSPIR]*

Joachim Jacobs, Sprecher der FSFE, bringt in diesem Zitat eine der wichtigsten Überlegungen zu diesem Thema zur Sprache. Eine Vorreiterrolle in diesen Belangen nimmt das BG Rechte Kremszeile ein, welches seinen Schülern idente Kopien des in der Schule eingesetzten Betriebssystems inklusive aller Anwendungssoftware anbietet. Ein Service, der durch die dritte postulierte Freiheit der FSF - die Freiheit, Kopien weiterzugeben, um seinen Mitmenschen zu helfen - überhaupt erst ermöglicht wird. Der Vorteil dieser Vorgehensweise liegt auf der Hand. Die Schüler können zu Hause mit der selben Software arbeiten, die sie aus der Schule gewöhnt sind und müssen hierbei weder finanzielle noch rechtliche Faktoren berücksichtigen. Die speziell angepasste Linux-Distribution „LinuxAdvanced“, auf die ich im Abschnitt 4.7 „Einsatz von freier Software in Unterricht, Verwaltung und Infrastruktur“ noch zu sprechen kommen werde, steht auf der schuleigenen Webseite zum Download zur Verfügung und bietet somit auch anderen Schulen die einzigartige Chance, von der investierten Arbeit des LinuxAdvanced-Teams zu profitieren und diese für ihre Zwecke einzusetzen.

Projekte dieser Art, die dem kooperativen Modell folgen, gibt es mehrere und sie erweisen sich oft als wertvoller für öffentliche Organisationen als andere konkurrierende Systeme.

> *Hier zeigt sich ein weiterer Vorteil von quelloffener Software, deren bekanntester Vertreter ja Linux ist. Das zu Grunde liegende, frei zugängliche Betriebssystem wurde genommen, ebenso die anderen benötigten Komponenten, um in freiwilliger Arbeit ein auf Schulzwecke abgestimmtes Serversystem zu erstellen. Zum großen Teil von Lehrern für Lehrer. Auf dieser Basis entstanden der genannte c't-ODS-Server, der zwar von der Zeitschrift c't ursprünglich initiiert und bis heute vertrieben, aber von Lehrern weiter entwickelt wird. Ein weiteres Beispiel ist der an der Gesamtschule Eiserfeld entwickelte GEE-Server. Diese Systeme, genau wie ihre kommerziell erstellten Konkurrenzprodukte Easy-Admin und ScioLiner, bieten neben hohem Komfort und Flexibilität eine Funktionalität, die eine weitere Schulverwaltungs-Software in vielen Bereichen überflüssig macht. Und ohne die Möglichkeit, das System zu studieren und zu verändern, wären diese Lösungen zu diesem Preis nicht entstanden.*
> *[BING,S.4]*

4.5 Sicherheit und Free Open Source Software

Freie Software per se ist keine Sicherheitsstrategie, bietet jedoch eine ausgezeichnete Basis und damit bedeutende strategische Vorteile.

„Vertrauen versus Wissen.“
„Viele Augen sehen mehr als wenige.“

So und so ähnlich lauten die Argumente der Vertreter der freien, quelloffenen Software, welche nicht so einfach von der Hand zu weisen sind. Fakt ist, geschlossener Quellcode zentriert die Frage der Sicherheit auf die Hersteller und hinterlässt die Konsumenten in diesem Punkt als unmündig-passive Benutzer der Software. Ein sicherheitstechnisch miserabel programmiertes Werkzeug kann jedoch von der Tatsache profitieren, dass Fehler nicht sofort und für jeden zugänglich sind. „security by obscurity“, wie dies in Fachkreisen genannt wird, sollte aber nicht der ausschlaggebende Grund für die Verschlüsselung des Quellcodes sein und spielt bei professioneller Software mit hoher Wahrscheinlichkeit keine relevante Rolle.

„[...] es gibt in der Kryptographie das „kerkhoffsche“ Prinzip, das sagt, ein Verschlüsselungsalgorithmus soll so konzipiert sein: Die Sicherheit darf nicht am Algorithmus hängen. Der muss publiziert sein. Die Sicherheit darf einzig und allein am Schlüssel hängen. Im Prinzip, wenn du nur nicht weißt, wie verschlüsselt wurde, ist das praktisch „security by obscurity“, weil die Sicherheit nicht in dem Schlüssel steckt. Bei allen vernünftigen Standards wie AES z.B. sieht es folgendermaßen aus: jeder Mensch kennt den Algorithmus, jeder Mensch kann ihn lesen, aber er kennt den jeweiligen Schlüssel nicht, also kann er es nicht entschlüsseln und so denke ich, ist es auch mit der Sicherheit von Software. Nur weil ich nicht weiß, wo der Fehler von einem Programm ist, heißt das nicht, dass ich ihn nicht angreifen kann. Also da bin ich sicher auf der Seite der Open Source, weil es einfach möglich ist, das zu sehen. Es bietet vielleicht auch mehr Angriffsflächen, die aber schneller kleiner werden.“
[Florian Wisser, Interview 2008]

Wie Mag. Florian Wisser in diesem Interview erwähnt hat, bietet offener Quellcode den geneigten Angreifern natürlich auch die praktische Möglichkeit, diesen zu studieren und eventuelle Fehler und Angriffspunkte zu erkennen. Die Praxis jedoch zeigt, dass sich dies nicht zum Nachteil quelloffener Software auswirkt. Durch das Release-Schema „release early, release often“, wird freie Software in der Regel schon im frühen Alpha-Stadium veröffentlicht und von der Community eingesetzt und getestet. Der Sinn von „release early“ ergibt sich durch die hohe Anzahl an freiwilligen „Beta-Testern“ die eine rasche Beseitigung von Fehlern und somit eine verhältnismäßig ausgereifte Final-Release ermöglicht. Eventuelle Fehler werden in der Praxis schnell gefunden und, da dies durch die Freiheiten der Software gewährleistet wird, gefixt und sogenannte „Patches“ veröffentlicht und an die Community zurückgegeben.

„[...] wenn ich sage, der Code ist offen, kann es sein, dass hier jemand etwas einbaut, es wieder ins Internet stellt und ich das dann herunterlade und dadurch ein Sicherheitsproblem bekomme, ohne es überhaupt zu wissen; das ist sicher ein gewisses Risiko.“ [Marco Delbello, Interview 2008]

MMag. Delbello verweist hier auf die Möglichkeit, dass solche „Patches“ nicht immer der Verbesserung der Software dienen müssen, sondern auch Probleme verursachen können. Die Idee, auf diese Weise ein Sicherheitsleck zu verursachen, ähnelt jener des „phishings“ und stellt durchaus ein Problem dar, bedarf allerdings eines aktiven Benutzers, welcher sein eigenes System kompromittiert. Zu diesem Zweck muss die geschützte Umgebung gewarteter Paketquellen absichtlich verlassen und unter Angabe des root-Passworts eine schädliche Fremdquelle genutzt oder ein schädlicher „Patch“ heruntergeladen und installiert werden. Aus diesem Grund definiert sich dieser Aspekt auch nicht direkt als Sicherheitsmangel der Software und lässt sich nur schwer auf die Offenheit des Quellcodes zurückführen. Letztlich bietet hier offen gelegter Quellcode erneut den Vorteil gegenüber geschlossenem Code, diese schadhaften Veränderungen überhaupt zu entdecken, wie es MMag. Delbello in folgendem Zitat selbst ausdrückt.

> *„Der Vorteil ist natürlich, je mehr Leute sich den Code ansehen, desto sicherer wird auch die Software.“ [Marco Delbello, Interview 2008]*

Das BSI (Bundesamt für Sicherheit in der Informationstechnik) schreibt hierzu:

> *Beim Einsatz der Freien Software sind dem BSI folgende technische Aspekte besonders wichtig:*
>
> - *Warnmeldungen über bei Sicherheitsprüfungen gefundene Fehler können veröffentlicht werden, weil es kein Non Disclosure Agreement gibt. Der Anwender kann so bei Sicherheitslücken schnell informiert werden und Gegenmaßnahmen ergreifen.*
> - *Die Prüfung von Software auf Sicherheitslücken sollte immer möglich sein. Beim Einsatz von Software kann dies ein K.O.-Kriterium sein. Es steht Vertrauen versus Wissen.*
>
> *[BSI]*

An dieser Stelle muss jedoch festgehalten werden, dass freie Software allein keine Sicherheitsstrategie darstellt.

Die Unix-artige und sehr ausgereifte Rechteverwaltung eines GNU/Linux Systems bedeutet im Anwendungsalltag zwar eine hohe Resistenz gegen Angriffe und vor allem gegen Viren und moderne Netzwerkviren (Würmer), welche u.a. aus diesem Grund für derartige Systeme praktisch nicht existieren; ein gut gewartetes System, komplexe, sichere Passwortvergabe, regelmäßige Sicherheitsupdates sowie ein gut geplantes System, in dem Dienste mit eingeschränkten Rechten laufen und komplexe Paketfilter zum Einsatz kommen, ist jedoch unabdingbar. Ein hohes Risiko ist und bleibt letztlich der Faktor Mensch. Offene Terminal-Sessions sollten aus diesem Grund nach vordefinierter Idle-time automatisch schließen und der Schulserver für nicht autorisierte Personen unzugänglich sein. Ich schließe diesen Abschnitt daher mit dem „Fazit“ des BSI:

> *„Unabhängigkeit und Software-Vielfalt sowie die Verwendung offener Standards bieten eine Basis für IT-Sicherheit. Sicherheit ist jedoch ein Prozess. Um IT-Sicherheit erhalten zu können, müssen die Verantwortlichen das System genau kennen, regelmäßig warten und Sicherheitslücken schnell beheben. Der Einsatz Freier Software bietet per se keine Gewähr für ein sicheres System. Er bietet in diesem Prozess jedoch bedeutende strategische Vorteile.“ [BSI]*

4.6 Open Content im Unterricht

Wikipedia und andere Open Content Plattformen sind aus heutiger Sicht nicht mehr aus dem Unterricht wegzudenken. Umso wichtiger erscheint es daher, den Umgang mit diesen Informationsquellen und den Diskurs über die Problematik der Qualitätssicherung aktiv in diesen einzubinden.

Die Idee der grenzenlosen Freiheit des Wissens bzw. generell von Inhalten geht über das viel diskutierte Thema der Software und deren (Un-)Freiheit hinaus. Während die Freiheit bei Software nach wie vor Definitionssache zu sein scheint und die Vertreter proprietärer Software, wegen ihrer strikten Definition von Software als Produkt mit Marktwert, durchaus gleichzeitig für freie Bildung und eine freie Wissensgesellschaft sein können, ohne widersprüchlich zu handeln, sieht dies bei Wissen etwas anders aus.

Wissen, welches bereits Allgemeingut ist und vom moralischen Standpunkt aus jedem zugänglich sein soll, wird dennoch in Enzyklopädien oder ähnlichen Sammelwerken wie Wörterbüchern und diversen Nachschlagewerken vertrieben. Brockhaus, Langenscheidt, aber auch Microsoft mit der Software-Enzyklopädie „Encarta“ vermarkten dieses Wissen bereits seit Jahrzehnten auf herkömmliche Weise und verlangen, zu Recht, für die Zusammenstellung und die Aufbereitung des Wissens sowie natürlich für das Trägermaterial, den Druck bzw. die Herstellung eine angemessene Kostenentschädigung. Heute fällt die Wahl bereits häufig auf die digitale Version der Produkte, da diese sich oft wesentlich funktionaler einsetzen lässt und nicht zuletzt kaum Platz in Anspruch nimmt.

Die Revolution des Internets ist gerade in diesem Bereich kaum aufzuhalten und es stellt sich die Frage, welche Existenzberechtigung derartige Wissenssammlungen in einer freien Gesellschaft noch haben; ob Wissen generell patentierbar sein sollte und ob ein „Copyright“ auf Texte aus dem Brockhaus in fünfzehn Bänden dem Sinn der Entdecker dieses Wissens entspricht. *Wikipedia*, *Wikimedia Commons*, *Wiktionary*, *Wikibooks*, *Openthesaurus*, *Pauker*, *Wortschatzlexikon* und viele andere großartige Projekte dieser Art behaupten das Gegenteil.

Ein „Wiki“ beschreibt eine spezielle Sorte von Webangeboten, die es den Benutzern ermöglichen, schnell und ohne technische Vorkenntnisse die Inhalte online direkt im Webbrowser zu editieren. In der Regel handelt es sich hierbei um Gemeinschaftsprojekte, in denen Inhalte gesammelt und aufbereitet werden. Im Allgemeinen ist es üblich, die verschiedenartigen Inhalte, angefangen von textlicher Information über Bilder bis hin zu Musik und Videos kurz als „Content“ zu bezeichnen. Im konkreten Falle sprechen wir von Open Content, wenn diese Inhalte ohne Abgabe von Lizenzgebühren verbreitet, bearbeitet und gewerblich genutzt werden dürfen und somit diametral zu Werken stehen, denen das Urheberrecht eine solche Nutzung verbietet. Als Analogon zur Freien Open Source Software entstand der Begriff des „Open Content“ erst 1998, geprägt durch die Open Content Initiative von David Wiley.

Diese gemeinschaftlichen Sammlungen von Wissen aus aller Welt halten verständlicher Weise auch in Schulen seit geraumer Zeit Einzug und beginnen zusehends Schulbibliotheken als ultimative Referenz für Schüler zu ersetzen. Sie liefern schnelle, vernetzte Resultate und stellen somit die Basis so mancher Schülerreferate, Hausübungen und Projekte dar. Qualitätssicherung ist jedoch nach wie vor ein großes Problem für derartige Projekte. Aus diesem Grund sollte in der Schule über diesen Umstand aufgeklärt und die Evaluierung dieses Wissens auch im Unterricht thematisiert werden. Das Wikipedia Schulprojekt, welches erstmals im September 2006 am Hildesheimer Gymnasium Adreanum stattfand, setzte sich das Ziel, diese Problematik in der Schule zu thematisieren und bot Workshops für Schüler, Lehrer und interessierte Eltern an. Auch Projektunterricht, in welchem z.B. Wikipedia Artikel gemeinsam erstellt, erweitert und evaluiert werden, um Schüler tiefer in die Materie einzuführen, wäre zudem denkbar und könnte auf lange Sicht neben Lernerfolg auch zu erhöhter Qualität des vorhandenen, öffentlichen Wissens führen.

4.7 Einsatz von freier Software in Unterricht, Verwaltung und Infrastruktur

Da sich freie Software ausgezeichnet für den Einsatz im Unterricht eignet und den Wartungsaufwand langfristig zu senken vermag, gehört es zur sozialen Verantwortung einer allgemeinbildenden höheren Schule, diese zumindest in Betracht zu ziehen und für ihre spezielle Situation zu prüfen, wie und in welchen Bereichen diese am besten zum Einsatz kommen kann.

> *„For some of the participants, it was clear that OSS was chosen not only to save money, but because it was seen as embodying and helping to implement a collaborative ethos.“*
> *[BECTA, S.14]*

Die Schaffung und Stärkung des gemeinschaftlichen Bewusstseins ist nur eine von zahlreichen Prämissen, die bei der Entscheidung freie Software einzusetzen zum tragen kommen. Im Gegensatz zur Wahl des Betriebssystems stehen bei der Wahl der Anwendungssoftware auch pädagogische Aspekte vermehrt im Vordergrund. Wie schon in einem früheren Abschnitt erwähnt, sollte in der Schule das Erlernen von Konzepten anstelle von Programmen das deklarierte Ziel des (Informatik-)Unterrichts sein.

> *„Ich denke, eines der wichtigsten Dinge wäre es, den Leuten Grundfunktionalitäten und Grundkonzepte zu zeigen; also wenn ich in einem Browser in ein Fenster hinein tippe, was kann das alles sein, was ist https im Unterschied zu http, solche Sachen, also wirklich grundlegende Dinge, um den Leuten zu zeigen, ist das plausibel, was da steht; kann das technisch überhaupt stimmen oder will mich da wer auf den Arm nehmen? Die Leute sind natürlich unsicher in der Frage, denn sie wissen ja nicht, was sie tun. Dies ein bisschen zu stärken wäre eine wichtige Aufgabe. Ich finde, konkrete Ausbildung auf eine bestimmte Software ist nicht Aufgabe der Schule, eher allgemein zu bilden.“*
> *[Florian Wisser, Interview 2008]*

Das Vermitteln von Heuristiken und Lösungen, die Kenntnis von übergreifenden Konzepten hat im Gegensatz zu dem bereits ein Jahr später obsoleten Spezialwissen über die Bedienung einer spezifischen Anwendungssoftware einen weitaus nachhaltigeren Effekt. Führt man sich beispielsweise die Bedienung von Textverarbeitungsprogrammen vor Augen, so kann man zwar idente Konzepte erkennen, die jedoch, angefangen von Überschriften, Inhaltsverzeichnissen, Fließtext um eingebettete Grafiken etc. unterschiedlich zu handhaben sind, sodass einseitig angelerntes Wissen hier beim Wechsel des Programms oft schon zum Arbeitsstopp führen könnte, was jedoch bei ausreichender Vorbereitung schnell überwunden werden könnte. Dies wird wesentlich vereinfacht, wenn den Schülern die verschiedensten Implementierungen der Software zur freien Verfügung stehen. Die vollständig transparente Idee des Betriebssystem GNU/Linux sowie dessen Design kann in der Schule dazu genutzt werden, die Funktionsweise eines Betriebssystems sowie die Ansprüche, die an ein solches gestellt werden, zu vermitteln.

Im Rahmen freier quelloffener Software kann des Weiteren veranschaulicht werden, wie diese ursprünglich aus dem „Sourcecode“ mittels „Compiler“ erzeugt wurde; dass diese in vielen verschiedenen Versionen vorliegt; welche Fehler und welche Lösungsmöglichkeiten für diese existieren bzw. wie man an der Entwicklung mittels „Bugreports“ auch ohne Programmierkenntnisse teilhaben kann. All dies lässt sich im Unterricht direkt am Rechner nachvollziehen. Linus Torvalds schreibt zu diesem Thema:

> *„Tatsächlich waren Computer damals besser für Kinder geeignet: Sie waren weniger ausgereift als heute, und Grünschnäbel wie ich konnten unter der Haube daran herum basteln. Heutzutage ist es mit Computern wie mit Autos: Mit zunehmender Komplexität wurde es schwieriger, sie auseinander zu nehmen und wieder zusammen zu bauen und auf diese Weise zu lernen, was eigentlich dahinter steckt.“ [JFF, S.24]*

GNU/Linux bietet jedoch nicht nur ein transparentes, sondern auch ein hoch entwickeltes Systemdesign. Echtes Multitasking und modularer Aufbau schaffen eine hohe Systemstabilität, selbst unter Verwendung extrem instabiler Teilkomponenten. Das intelligente Design verhindert, dass einzelne Programme das gesamte System gefährden können. Eine markante Eigenschaft eines GNU/Linux Systems - von erheblicher Relevanz für Schulen - ist die Unterstützung von Multi-User-Systemen, einer der zentralen Funktionen von GNU/Linux. Die Möglichkeit, eine offene Anzahl unabhängiger Benutzer auf einem Gerät zu verwalten, ohne dass diese in irgendeiner denkbaren Weise interferieren, bietet im Unterricht die Chance, völlig neue Wege zu gehen. Ein zentraler Nebeneffekt der guten Benutzerverwaltung ist die erhöhte Systemsicherheit. Auch in aktuellen Windows-Vista-Versionen wurde aus diesem Grund die Funktionalität in diesem Bereich überarbeitet. Ein Aspekt in dem sich Linux profilieren kann ist, dass sämtliche Änderungen, die ein Benutzer vornimmt, auf das eigene $HOME-Verzeichnis beschränkt bleiben. Aufgrund einer komplexen Rechtevergabe ist es gewöhnlichen Benutzern unmöglich, für die Systemstabilität relevante Teile des Betriebssystems zu verändern.

> *„Ich brauchte keine Angst zu haben, dass die Schüler etwas kaputt machen könnten. Unsere Computer haben alle keine Wächterkarten. Was der/die Benutzer/in verändert oder zerstört,was eingerichtet oder installiert wird, was verändert, zerstört oder installiert wird, das alles passiert bei Linux ausnahmslos im eigenen privaten Verzeichnis. Und das kann man jederzeit wieder neu einrichten.“ [BING, S.1]*

Vorhin hab ich den Schülern gesagt, sie können auf Ubuntu alles ausprobieren und ich hab das dann über das Gconf-Tool ganz einfach wieder zurückgesetzt. Ich hab gesagt, sie können Sachen hinzufügen, irgendwelche Icons und sonst was kaputt machen; mit einem kleinen Konsolenbefehl kann ich das alles wieder zurücksetzen. [Marco Delbello, Interview 2008]

Alle individuellen Konfigurationen liegen ausschließlich im $HOME-Verzeichnis der Schüler, was darüber hinaus auch für ihre privaten Daten gilt, welche ihnen in einem entsprechend konfigurierten Client-Server-System - völlig unabhängig davon, auf welchem Gerät sie sich einloggen - zur freien Verfügung stehen. So bietet dieses Systemdesign die Option, innerhalb des eigenen Accounts alles nach individuellen Bedürfnissen einzustellen, angefangen mit dem Layout der grafischen Oberfläche, Icons, Schriftstil, Schriftgröße, etc. bis hin zur Systemsprache und in Verbindung mit dem Benutzeraccount für spätere Zugriffe auf diesen permanent zu sichern, während auf anderen Systemen zu restriktiven Maßnahmen gegriffen werden muss, wie man dem folgenden Zitat entnehmen kann, um den administrativen Aufwand in einem vertretbaren Rahmen zu halten.

„ [...] da haben die Studenten aber ein unveränderbares Minimalprofil, das ist Server diktiert und um das so zu machen, ist es unter Windows am einfachsten, einen Windowsserver hinzustellen, dann gibt es diese Policyschalter, und man kann sagen: der sieht jetzt seine Dosbox nicht, usw. es war zwar nicht ganz trivial und zum Teil auch ein wenig idiotisch, aber die Einschränkungsmöglichkeiten waren gut.“ [Florian Wisser, Interview 2008]

Am Beispiel einer Hauptschulklasse zeigt Peter Bingel, wie die Flexibilität eines Systems für den Unterricht sinnvoll eingesetzt werden kann. Im folgenden Zitat bezieht er sich hauptsächlich auf die Umgebungssprache des Systems und führt dabei die Möglichkeit an, den Umgang mit Fachausdrücken zu vereinfachen und diese dadurch leichter zu erlernen.

„Simin spricht einen afghanischen Dialekt. Den kann ich ihr nun wirklich nicht anbieten, aber zum Glück versteht sie etwas englisch. So stellen wir mit einem Mausklick die Umgebungssprache auf englisch – und schon erscheint das nächste Programm mit englischsprachigen Menüs. Simins Gesicht hellt sich auf. Was sie nun auf dem Bildschirm sieht, kann sie wenigstens teilweise entziffern.“ [BING, S.2]

„Häufig fehlt auch das Verständnis für die ihnen unbekannten Fachausdrücke. Etliche Kinder fahren ihre Oberfläche in der ihnen vertrauten Mutter- oder Zweitsprache hoch. Mit dem Deutschen kommen sie zwar nach einiger Zeit in den Grundbegriffen klar, aber Fachtermini sind eben etwas anderes. So laufen türkische, russische, griechische, italienische, englische und spanische Bildschirmmenüs nebeneinander. Sogar ein chinesischer Bildschirm ist zu sehen – was ich bisher noch vermisse, ist eine bessere Unterstützung für arabische Sprachen. Aber auch die kommt bald. Langsam und vorsichtig zeige ich im Unterricht mit dem Beamer, welches Programm ich starten und was ich damit tun will. Langsam und vorsichtig folgten sie dem, was sie da vorne sehen, auf ihrem Monitor. Und während ich die einzelnen angeklickten deutschen Begriffe ihnen vorlese, können sie diese in der ihnen verständlichsten Sprache nachvollziehen.“ [BING, S.2]

Jetzt könnte man natürlich dagegen argumentieren und behaupten, dies würde den Lernprozess der deutschen Sprache beeinträchtigen. Nach eingehender Überlegung komme ich jedoch zu dem Schluss, dass dieser durch die Möglichkeit des schnellen Wechsels der Umgebungssprache jedoch eher gestützt würde. Im Informatikunterricht liegt das Lernziel in Fertigkeiten mit dem Umgang des PCs. Parallel zu diesem lernen die Schüler die deutschen Fachausdrücke kennen und können diese - nicht zuletzt auf Grund ihrer identen Positionierung in der Anwendungssoftware - in der jeweils anderen Sprache wiedererkennen. Der Sprachwechsel im Deutschunterricht z.B. geschieht mit wenigen „Klicks“, eine Funktion, die auch im Sprachunterricht von Vorteil sein kann.

„Und einige unserer deutschen Schüler wollen partout ihre Menüs in englisch einstellen. Sie betrachten dies als Vorteil für den Fremdsprachenunterricht und auch für das Stöbern im Internet.“ [BING, S.3]

Ein weiterer Vorteil im Systemdesign zeigt sich vor allem auch in der Administrierbarkeit. GNU/Linux Systeme bieten hier tatsächlich mehr als einen Werte- und einen Kostenvorsprung. Die bereits erwähnten $HOME-Verzeichnisse der Anwender liegen in einem Client-Server-System auf einem zentralen Server und werden während des Bootvorganges in der Regel von diesem per NFS geholt. Sämtliche Änderungen, die in diesem getätigt werden, geschehen somit auf dem Server.

Dies ermöglicht nicht nur eine flexiblere Handhabung der Clients, da die Daten nicht mehr an diese gebunden sind, sondern auch einen enormen Vorteil bei der Sicherung der Daten, die nun allein über den Server geschehen kann.

> *„Es ist zum Beispiel so, wenn ein Linux Computer eingeht, dann stellst du ihn weg, stellst einen neuen PC her, sagst ihm, welcher Client er werden soll, drehst ihn auf, er installiert sich, der User loggt sich ein und es hat sich für ihn nichts geändert. [...] Außerdem ist remote Administration von einem Linuxsystem viel einfacher. Die Administration macht sehr viel aus. Ich glaube auch, wenn man es mit einer konsequenten Politik macht, im Sinne von, das gibt es und wenn wir anfangen, Extrawurst zu erfinden, dann bauen wir es wirklich in das System ein, sodass es auch automatisch wieder installiert würde; wenn man das konsequent macht, hat man irgendwann wirklich viel weniger Arbeit.“ [Florian Wisser, Interview 2008]*

Natürlich setzen derartige Vorgänge, wie sie Mag. Florian Wisser beschreibt, eine entsprechend durchdachte und konfigurierte Umgebung voraus. Wie im Interview erwähnt wurde, spielt auch der Faktor der „remote Administration“ eine nicht unwesentliche Rolle. Innerhalb des Schulnetzen würde dies zum Beispiel bedeuten, administrative Tätigkeiten an den einzelnen Clients im Informatiksaal durchführen zu können, während andere auf diesen arbeiten. Im Falle einer simplen Server-Client-Struktur würde ein simples „ssh *root@clientx*“ es erlauben, auf dem Client - von Schülern unbemerkt – mit Superuser-Rechten zu arbeiten und eventuell Software zu installieren, die im Unterricht gebraucht wird. Während SSH und NFS, um zwei Beispiele zu nennen, fixer Bestandteil von GNU/Linux sind, müssen diese auf vergleichbaren Systemen nachinstalliert und konfiguriert werden und bieten erfahrungsgemäß weniger komfortable Praktikabilität. Über die technischen Aspekte hinaus stellt die Wahl einer freien Software-Lösung für den Unterricht eine sehr umsichtige Herangehensweise unter Einbezug zukünftiger Entwicklungen dar.

> *„Vor allem wenn ich mir denke, dass die Anforderungen an die Eltern auch von finanzieller Seite sicher nicht rückläufig sind, sondern im Bildungssystem zunehmen; in 10-15 Jahren wird vielleicht jeder mit einem Netbook in der Schule sitzen, das nicht die Schule zahlt, sondern die Eltern. Es wird für sie nicht billiger, dann ist es schon eine Selbstverständlichkeit, dass ich nicht leichtfertig eine Lösung vorschlage, die einfach nicht die beste Lösung ist und finanzielle Belastungen schafft, die einfach nicht notwendig sind.“ [Rene Schwarzinger, Interview 2008]*

Schwarzinger zufolge liegt die Wahl der Software(-lizenz) im theoretischen Zuständigkeitsbereich der Schule, da diese Rücksicht auf alle Beteiligten nehmen sollte und durch diese Wahl indirekt Einfluss auf Schüler und Eltern genommen wird. Für den Schulbereich findet sich bereits eine große Anzahl an spezialisierten Linux-Distributionen, deren Ziel es ist, die bestmögliche Lösung für ein komplexes Schulnetzwerk zu bieten sowie den Aufwand für die Administration von Server- und Desktopbereich möglichst gering zu halten. Der Nutzen einer solchen Zusammenstellung von Software, Skripten und fertigen Konfigurationen von Lehrern für Lehrer liegt auf der Hand: Arbeitsersparnis. Drei Vertreter dieser speziell angepassten GNU/Linux Distributionen werde ich im Folgenden in wenigen Worten beschreiben.

4.7.1 Linux Advanced

Abb. 7: Linux Advanced Logo [BGK]

Linux Advanced wird von einem sehr kleinen Team am BG Rechte Kremszeile entwickelt und richtet sich speziell an die Bedürfnisse der eigenen Schule und der Schüler. Anders als das Pendant „Skolelinux“ ist das Ziel dieser Distribution nicht, das Maximum an Möglichkeiten zu bieten. Sinn und Zweck ist es, das optimal an die eigenen Bedürfnisse angepasste Linux auch auf den privaten PC's der Schüler nutzen zu können. Der universelle Einsatz dieser Distribution steht u.a. im Vordergrund, weshalb LinuxAdvanced als Live-System entwickelt wird und daher auch ohne Installation direkt eingesetzt werden kann. Über die schuleigene Webseite können Images der Distribution für drei verschiedene USB-Stick Größen als auch für CD und DVD herunter geladen werden. LinuxAdvanced basiert auf einem Debian-System, das auf Grund der Langlebigkeit, Sicherheit und Stabilität, welche man diesem nachsagt, gewählt wurde. Um den ressourcenschonenden Einsatz des Systems zu ermöglichen, setzt man bei dieser Distribution auf Xfce als „lightweight“- Desktopumgebung. Eine detailliertere Beschreibung dieses Projektes lässt sich auf der Webseite des Gymnasiums sowie im Kapitel 5 „Modellbeispiel: BG Rechte Kremszeile“ nachlesen.

4.7.2 Skolelinux

Abb. 8: Skole Linux Logo [SKL]

Ähnlich wie LinuxAdvanced basiert auch Skolelinux auf Debian. Skolelinux, ein wichtiger und weit verbreiteter Vertreter spezieller Schuldistributionen, bietet zudem einfache Installationsmöglichkeiten für Server, Workstation, Terminalserver und Thinclients und setzt dabei auf KDE als Desktopumgebung. Alle Benutzer des GNU/Linux-Systems werden zentral im LDAP verwaltet, auch werden alle Nutzerdaten zentral gespeichert, sodass unabhängig davon, an welchem Client gearbeitet wird, die persönlichen Desktopeinstellungen vorgefunden werden. Wie auch bei LinuxAdvanced wird von einfacher Konfiguration und Verwaltung von Profilen bis hin zu automatischen Backups alles in dieser Distribution zusammengefasst und einfach aufbereitet angeboten. So erlaubt ein leicht zu konfigurierender Proxy-Server, Inhalte des WWW zu filtern und den Zugriff auf das Internet zu verwalten.

4.7.3 Edubuntu

Abb. 9: Edubuntu Linux Logo [EDU]

Diesen bekannten Vertreter unter den Schulversionen von GNU/Linux, „Edubuntu“, darf man an dieser Stelle nicht vergessen. Edubuntu versteht sich in erster Linie als Erweiterung zu Ubuntu, von dessen starker Verbreitung es profitiert. Es definiert sich durch vereinfachte Benutzbarkeit des Desktops und einer sorgfältig getroffenen, aber kleinen Auswahl der Lernsoftware. Basierend auf Debian setzt diese Version auf Gnome als Desktopumgebung und bietet u.a. erste einsatzfähige Teile von SchoolTool an, welches im April 2009 released werden soll. SchoolTool ist ein Projekt zur Entwicklung eines Web-basierenden, einheitlichen, globalen Schuladministrationswerkzeugs und besteht zum derzeitigen Entwicklungsstand aus einem Kalendersystem und einer Schülerverwaltung.

4.7.4 (Informatik)Unterricht

Die folgende Liste soll einen Überblick über das Angebot an Anwendungssoftware für den Schuleinsatz schaffen und erhebt keinen Anspruch auf Vollständigkeit.

Programmname	***Kurzbeschreibung***
Qcad	*Technisches Zeichnen*
Gcompris	*Umfangreiche Lernsoftwaresuite für 2 – 10 Jährige*
TuxTyping	*Spielerisch die Tastatur kennen lernen*
Tuxpaint	*Ein sehr einfach gehaltenes Malprogramm für Kinder*
TuxMathScrabble	*Scrabble mit Zahlen und mathematischen Operanden*
Vym	*Mindmapper – erlaubt es Gedanken hierarchisch zu strukturieren*
KwordQuiz	*Lernkartei – nicht nur Vokabel lassen sich durch unterschiedliche Abfragemodi einstudieren*
Keduca	*Erlaubt die Erstellung von Lernkarteien und digitalen Fragekatalogen / Tests*
Jclic	*Autorensystem um interaktive Lerninhalte zu erzeugen*
KVocTrain	*Komfortabler Vokabeltrainer*
Klatin	*Lateinische Grammatik und Vokabel lernen*
Ktouch	*Intelligenter Tipptrainer*
(K)Octave	*Ein Programm zur Lösung von numerischen mathematischen Aufgabestellungen. (weitgehend zu Matlab kompatibel)*
QtiPlot	*Funktions- und Datenplotter*
Kalgebra	*Mathematiksoftware zum Zeichnen von Funktionen in 2D und 3D*
Kmplot	*Funktionsplotter für kartesische und polare Koordinatensysteme*
KBruch	*Einstudieren von Regeln der Bruchrechnung*
KPercentage	*Lernprogramm für Prozentrechnung*
Kig	*Interaktives Geometrie-Programm*
KCalc	*Virtueller Taschenrechner*
GeoGebra	*Speziell für den Mathematikunterricht konzipiert. Berechnung von Ableitungen, Integralen, etc.*
K3DSurf	*Erstellen von mehrdimensionalen Oberflächen mit Hilfe mathematischer Gleichungen. Schnittstelle zu PovRay (rendern&modellieren)*
KTurtle	*Logo für KDE*
Umbrello	*Schulung von UML Notation*
Alice	*3D Authoring für Anfänger*
Kalzium	*Interaktives Periodensystem der Elemente*

Step	*Interaktive virtuelle Umgebung für naturwissenschaftliche Versuche*
KSimus	*Konstruktion virtueller Schaltungen. Schaltpläne können interaktiv getestet werden.*
KGeography	*Umfangreiches Erdkunde-Lernprogramm*
Sunclock	*Weltzeituhr und Weltkarte*
Scret	*Notenlesen anhand zufälliger Melodien lernen*
Kanagram	*Anagramme erraten*
KHangman	*Klassisches „hangman“ Spiel zum erlernen von Wörtern in einer Fremdsprache*
Kiten	*Japanisch lernen*
KLettres	*Buchstaben lernen*
Parley	*Umfangreiches Lernprogramm u.a. für Vokabel und andere Inhalte*
Blinken	*„Simon Says“ mit Farben. (Gedächtnistrainer)*
KVerbos	*Spanisch konjugieren*
Keduca	*Erstellung digitaler Tests*
Kstars	*Desktopplanetarium*
Marble	*3D Weltatlas*
OpenOffice	*Textverarbeitung*
Gimp	*Fortgeschrittene Bildbearbeitung*
Hydrogen	*Komplexe „Drum Machine“ mit Song Editor*
Ardour	*Digitale mehrspurige Audiobearbeitung*

Wie man in der Liste an dem häufig vorangestellten K (zumeist bei KDE spezifischer Software) erkennen kann, werden eine große Anzahl an Lernprogrammen im Rahmen des KDE-Projekts entwickelt, dem „KDE Education Project“, welches konkret in der neuesten Version folgende Softwaresammlung (Illustration 10) beinhaltet.

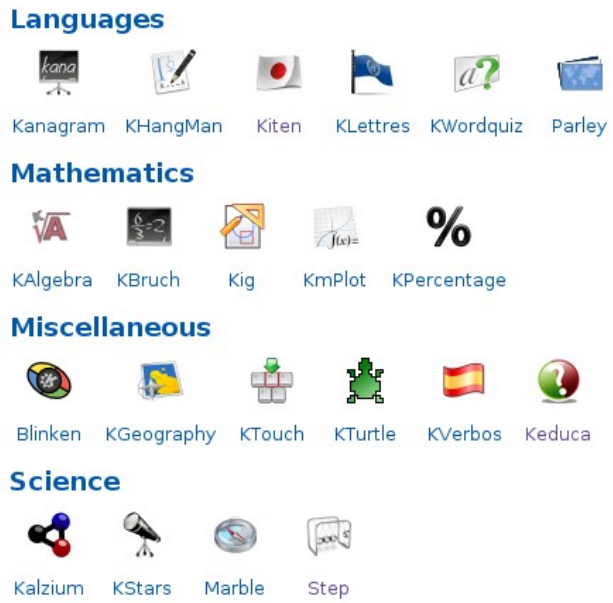

Abb. 10: KDE Educational Software - Überblick

4.7.5 Verwaltung

In der Schulverwaltung eingesetzte Software ist zu einem hohen Anteil nicht ohne erheblichen Aufwand durch freie Software zu ersetzen. Projekte wie WINE müssen in Erwägung gezogen werden, um Windowssoftware auf einem Linuxsystem zu betreiben bzw. andere Wege einer sanften Migration respektive einer Teilmigration. Je nach Softwareausstattung müssen hier neue Möglichkeiten erdacht werden, die sowohl eine Kombination aus Linux als Betriebssystem in Verbindung mit windowsspezifischer Spezialsoftware, betrieben unter WINE, als auch OpenOffice.org, Thunderbird und Firefox etc. auf einem Windowssystem bedeuten kann. Auch Windows als Basis zu behalten, langsam und schrittweise auf plattformunabhängige Software zu wechseln und insbesondere bei zukünftigen Ausschreibungen darauf zu achten, nur Software zu erwerben, die den heutigen Anforderungen entsprechend plattformunabhängig betrieben werden kann, wäre eine empfehlenswerte Herangehensweise, die vor allem auch vom Ministerium unterstützt werden sollte, welches im derzeitigen Modell die Lizenzen für die Schulen erwirbt.

4.7.6 Kooperation/Infrastruktur

Moodle, Joomla, Apache, BIND, MySQL, LDAP, NFS, Samba, Proftp, Postfix, OpenExchange, um nur einige wenige Beispiele für Vertreter aus der FOSS-Welt zu nennen, bieten vom kooperativen Arbeiten über ein freies E-Learning System, über Webseiten-Management mit einem modernen freien Contentmanagmentsystem (CMS) bis hin zu allen notwendigen Serverdiensten und Benutzerverwaltung professionellen Ersatz für alle gängigen Systeme. Um jedoch eine Windowsclient-Umgebung effizient zu verwalten, wird man um einen Windows-Domaincontroller nicht ohne Aufwand herum kommen. Unter anderem ist aus diesem Grund bei einer Teilmigration immer mit spezifischen Schwierigkeiten zu rechen, die in einer homogenen Server-Client Landschaft zumeist vermieden werden können.

4.8 Hindernisse einer (Teil)Migration

Gutes Ressourcen-Management, sowie die volle Unterstützung seitens der Kollegenschaft und der Eltern ermöglichen eine erfolgreiche Migration.

Die Umstellung der schulinternen, technischen Infrastruktur auf freie Software stellt sich schnell als schwieriges Unterfangen heraus, wenn in Schulen mit einem sehr kleinen Netzwerk und nur einem Server eine Systemumstellung der IT-Infrastruktur geplant ist. Auf Grund des notwendigerweise unterbrechungsfreien Betriebs bedarf es in dieser Situation eines temporären Ersatzservers. Weitaus besser situiert zeigen sich Schulen mit mehreren Servern. In diesen können gewisse Bereiche des Netzes versuchsweise auf freie Software umgestellt und der permanente Betrieb des Systems durch Auslagerung gewisser Dienste während der Umstellung aufrecht erhalten werden. Wenn auch der potentielle Nutzen einer Migration auf freie Software diese zu einer überlegenswerten Option für jede Schule macht, so mag es dennoch nicht für jede einzelne Schule der optimale Weg sein und es bedarf vorsichtiger Planung und Berücksichtigung der speziellen Bedürfnissituation einer jeden Schule. Ein vollständiger Wechsel wird auf Grund von sich momentan im Einsatz befindlicher Spezialsoftware auch oftmals nicht möglich sein. Eine sanfte, jedoch aufwändige Strategie wäre es in diesem Falle, „dualboot“ Systeme aufzubauen bzw. auf eine Virtualisierungslösung zu setzen und nur neue Software als freie Software einzuführen, um das bestehende System langsam aber stetig umzustellen.

> *“(Head Teacher) New systems introduced into the school will run OSS – so its use will increase. The intention is only to use proprietary products when absolutely necessary.” [BECTA, S.14]*

Wie es der Direktor einer britischen Mittelschule ausdrückte, würde auch dies zum langfristigen Ziel der vollständigen Unabhängigkeit von proprietärer Software führen können, vorausgesetzt die alternative, freie Software sowie das Know-how dafür ist in der Schule vorhanden. Rene Schwarzinger, IT-Manager des BG Rechte Kremszeile, sieht die Schwierigkeiten jedoch weniger in technischer als vielmehr in menschlicher Hinsicht.

„Der wesentliche Punkt ist, die Unterstützung der Kollegenschaft zu kriegen und da muss man rechtzeitig anfangen, dann ist das kein Problem; aber man darf diesen Punkt nicht unterschätzen, das ist mindestens genauso wichtig, in die ganze Planung einzukalkulieren wie das Technische. [...]. Auch die Eltern darf man nicht vergessen; wir haben nachher auch Eltern informiert, zuerst bei Elternabenden und im Elternverein, also war auch hier die Rückendeckung vorhanden, und technisch: nicht zu spät anfangen, schauen, dass man auch die Ferien zum Testen hat, wo man letzte Troubles noch bereinigen kann. Natürlich ist es positiv, wenn in der Schule mehr als eine Person dafür verantwortlich ist, wenn sich mindestens zwei Leute finden, die dann auch Engagement zeigen, gewisse Sachen zu lösen; das wäre von Vorteil.“ [Rene Schwarzinger, Interview 2008]

Zusammenfassend könnte man sagen, dass die Rahmenbedingungen für einen Umstieg sorgsam kalkuliert und größtenteils selbst geschaffen werden müssen, dass das notwendige Know-how für eine Umstellung auf freie Software und die folgende Wartung derselben in der Schule vorhanden sein sollte und der für die Migration eingeräumte Zeitrahmen großzügig bemessen sein sollte.

5 Modellbeispiel: BG Rechte Kremszeile

Steckbrief:

BG Rechte Kremszeile

Schwerpunkt Informatik:

1Stunde in 1.-4. Klasse

2Stunden in 5.-8. Klasse

~ 50 Lehrer

~ 500 Schüler

~ 130 Computer

5.1 Werdegang des Systems

Die Idee zu LinuxAdvanced ist im Herbst 2006 entstanden. Initiiert wurde das Projekt von Dr. Klaus Misof, welcher es für eine gute Idee hielt, Infrastruktur und Unterricht der Schule auf Linuxbasis aufzuziehen. Gemeinsam mit MMag. Rene Schwarzinger - in einem sehr kleinen Team - war man sich schnell einig, dass es Sinn machen würde, wenn man die notwendigen Rahmenbedingungen schaffen könnte. Die anfänglich vermuteten Schwierigkeiten mit der Kollegenschaft im Zuge der Umsetzung blieben größtenteils aus, da die Nutzung der Informatiksäle seitens der anderen Unterrichtsfächer eher noch gering war, dem Projekt eine angemessene Vorlaufzeit von einem halben Jahr eingeräumt wurde und vor allem die Kollegen rechtzeitig informiert wurden.

> *„[...] sie haben es nicht nur akzeptiert, sie haben es wirklich unterstützt, geklatscht in der Konferenz und sie waren begeistert dafür. [Rene Schwarzinger, Interview 2008]*

Im Team kam man sehr schnell zu dem Entschluss, dass vorhandene Linux-Distributionen den gestellten Anforderungen nicht zu 100% entsprechen würden. Die wichtigste dieser Anforderungen war, dass diese Linux-Distribution nicht nur in der Schule eingesetzt werden kann, sondern auch von den Schülern zuhause, was die Benutzung eines Live-Systems nahe legte.

Zunächst wurde versucht, ein adaptierbares Grundsystem zu finden, das auch auf älterer Hardware läuft. Aus der Überlegung heraus, dass es ein sehr schlankes System sein sollte, beschloss man schließlich auf Xfce zu setzen, welcher im Gegensatz zu weit verbreiteten grafischen Oberflächen wie KDE oder Gnome ausgesprochen Ressourcen schonend arbeitet. Eine weitere Bedingung war ein System auf Debian-Basis, da für dieses eine enorme Anzahl von Softwarepaketen zur Verfügung steht und bei dieser Linuxversion die Sicherheit und Stabilität im Vordergrund steht. So kam man über Dreamlinux und Ubuntu zu der Einsicht, dass ein reines Debiansystem die beste Basis für das Schulprojekt sei, insbesondere das Debian „live“ Projekt, welches sich von anderen „live“ Distributionen auch dadurch abhebt, dass es annähernd gleich aufgebaut ist wie das installierte System. Zum Volleinsatz kam LinuxAdvanced zum ersten Mal im Jahr 2007 in allen Informatiksälen, im Konferenzzimmer sowie in den Spezialsälen; nur in der Verwaltung anfangs nicht, deren Umstellung aber mittlerweile fortgeschritten ist. Ein Schwerpunkt wurde auf OpenOffice gesetzt, welches schon vorher unter Windows installiert wurde, um den Umstieg zu erleichtern. Hinzu kamen laufend schulinterne Lehrerfortbildungskurse, noch bevor Linux auf den Schulrechnern zu sehen war, und im Bedarfsfall individuelle Betreuung von Kollegen. Bereits vor dem Sommer 2007 wurde der erste Linux-Kurs angeboten und gezeigt, wie das System aussieht, das im Herbst Einzug in die Schule hält und auf die Unterschiede zu dem gewohnten Windowssystem hingewiesen. Office Management und auch Bildbearbeitung mit „Gimp“ wurden unterrichtet. Obwohl die Kurse zumeist am späteren Nachmittag angesetzt wurden und somit die Freizeit der Kollegen in Anspruch nahmen, sind diese nach eigener Aussage relativ gut angenommen worden und durch die Fülle der Kurse konnte das für die Umstellung notwendige Wissen erfolgreich übertragen werden.

LinuxAdvanced wird ca. 2mal im Jahr released, auch wenn nach eigener Einschätzung ein jährlicher Releasezyklus ausreichend wäre. Insbesondere wäre darauf zu achten, dies vor den Sommerferien einzurichten, um Administratoren die notwendige Zeit zum Testen eines neuen Systems zu garantieren. Die Entwicklung des Systems wird von MMag. Rene Schwarzinger als zeitaufwändig, jedoch nur kurz vor einer Release als intensiv beschrieben. Darüber hinaus sei zu beachten, dass dieser Aufwand nur einmal geschehen muss und andere, die die Distribution nur einsetzen wollen, diesen nicht zu betreiben haben, was durch die gesicherte, legale Weitergabe von freier Software gewährleistet ist.

5.2 Ziele der Schule

LinuxAdvanced ist der Versuch, eine Softwarelösung zu entwickeln, mit der ein Großteil der Schulen im deutschsprachigen Raum problemlos und sinnvoll arbeiten kann; die nicht nur auf die Schule beschränkt ist, sondern auch von den Schülern zuhause eingesetzt werden kann. Aus diesem Grund liegt der Schwerpunkt der Entwicklung auf dem „live“ USB-Stick, einem voll funktionstüchtigen Betriebssystem inklusive aller Anwendungssoftware, lauffähig direkt vom USB-Stick ohne vorhergehende Installation. Der Vorteil eines solchen Systems liegt in der Mobilität der persönlichen Daten, der Wartungsfreiheit und auch der Geschwindigkeit gegenüber einem „live“ CD-System. Schüler müssen auch zuhause nichts mehr installieren und haben ihr individuelles System stets bei sich. Installiert wird das USB-Image vorerst nur jenen Schülern, die zuhause PC-Systeme in Betrieb haben, die nicht wesentlich älter als zwei Jahre sind, um Probleme mit dem Bootvorgang von einem USB-Device zu vermeiden.
Ein wichtiger Punkt bei der Wahl der Schulsoftware ist, dass die Software in jedem Fall plattformübergreifend zur Verfügung stehen muss und diese somit auch auf eventuellen Windows und Mac Systemen zuhause eingesetzt werden kann.

5.3 Software im Schuleinsatz

Das Betriebssystem ist ein Debianlinux System mit vorinstallierter grafischer Desktopumgebung Xfce. Benutzerverwaltung gewährleistet eine LDAP Datenbank, die $HOME Directories der Schüler liegen auf einem Fileserver, welcher die Authentifizierung am LDAP vornimmt und diese per NFS exportiert. In der Schule kommt für den Office Management Unterricht hauptsächlich Open Office zum Einsatz, Firefox als Webbrowser, Thunderbird als Mailclient, Audacity für die Audiobearbeitung und Gimp für Bildbearbeitung sowie VLC als Mediaplayer. Für den Englischunterricht gibt es die Lernsoftware „you and me“, die jedoch eher selten eingesetzt wird und ebenso wie eine spezielle Biologiesoftware, unter WINE (Wine Is Not an Emulator) betrieben wird. Als Tipptrainer z.B. kommt Ktouch zum Einsatz, der mit eigenen Kursmodulen bestückt wurde, damit dieser mit dem Lehrbuch übereinstimmt. Des weiteren findet man unter LinuxAdvanced noch Kpercentage, Kbruch, etc. .

5.4 LinuxAdvanced - Ausblick

Der Support des System wird voraussichtlich im Haus belassen. Die Bemühungen gehen in die Richtung, den Betrieb von LinuxAdvanced möglichst wartungsarm zu machen. Das nächste Ziel ist es, die Schüler auch in der Schule ausschließlich von ihrem eigenen USB-Stick starten zu lassen.

Das BG Rechte Kremszeile plant im folgenden Jahr, von installierten Systemen in den Informatiksälen komplett abzugehen und ausschließlich auf das USB-Linux zu setzen. Es sind USB-Ladestationen geplant, an denen die Schüler eventuell zerstörte Systeme binnen weniger Minuten wieder herstellen können, sowie zwei verschiedene Versionen von LinuxAdvanced für die Unterstufe und Oberstufe, deren Unterschied sich in erster Linie durch den Schreibschutz der Unterstufenversion zeigt, was eine lange Lebensdauer des Betriebssystems garantieren soll. Es wurden bereits die neuen Rechner ohne CDROM Laufwerk gekauft und auch Festplatten würden in Zukunft nicht mehr von Nöten sein, was in der Folge sowohl eine enorme Einsparung an Kosten für neue Rechner als auch bei den Wartungskosten selbst bedeutet. Es soll ein P2P Netzwerk entstehen, in dem jeder mit dem eigenen USB-Stick hochfährt und jeder Freigaben auf seinem System freischalten und auf andere zugreifen kann. Der Server wird soweit als möglich abgespeckt und der Mailserver ausgelagert. Der Schulserver soll in letzter Instanz nur noch die Webseite, Skripte für Internetsperre und Samba beherbergen und der USB-Stick somit zum zentralen Medium der Schule werden.

6 typeX-press – Freie Software in der Praxis

In vorliegendem Abschnitt wird das Projekt „typeX-press“ als Anschauungsbeispiel eines Free Open Source Projektes vorgestellt und die technischen Grundlagen desselben erläutert.

6.1 Konzept und Ziel der Software

Das Contentmanagmentsystem „typeX-press“ soll es in erster Linie sowohl Anfängern als auch Fortgeschrittenen ermöglichen, Inhalte direkt am Server zu bearbeiten und zu veröffentlichen. Ein bemerkenswertes Feature dieses CMS ist es, bereits bestehende Websites in das System aufnehmen zu können, ohne diese komplett portieren zu müssen. Die Zielgruppe dieser Software erstreckt sich vom Schüler bis zum Informatiklehrer und bleibt damit jedoch in erster Linie auf den Einsatz in der Schule beschränkt. Dies ergibt sich durch eine einseitige Ausrichtung bei den Systemhilfen und Vorlagen des CMS.

So können Module zur Sprechstundenverwaltung zum Beispiel natürlich auch an andere Einsatzszenarien adaptiert werden; die grundsätzlichen Überlegungen, die in dieses CMS einfließen, richten sich jedoch direkt an den Einsatz in einer Schule. Die Administration einer Schulwebsite soll in mindestens 3 Kategorien unterteilbar sein und somit die Vergabe verschiedener Rechte an Schüler, Kustoden oder Administratoren ermöglichen. Der zur Verfügung stehende Editor zum Bearbeiten einer einzelnen Seite bietet sowohl einen simplen WYSIWIG Editor als auch die Möglichkeit des Vollzugriffs auf den Quellcode einer Datei; ein bereits sehr ausgereifter Dateimanager ermöglicht die Verwaltung der Webseite und dessen Struktur - dank Ajax-Technologie so komfortabel wie eine lokal installierte Software. An den folgenden Abbildungen kann man sich einen oberflächlichen Eindruck verschaffen, wie die Software im derzeitigen Entwicklungsstadium gestaltet ist.

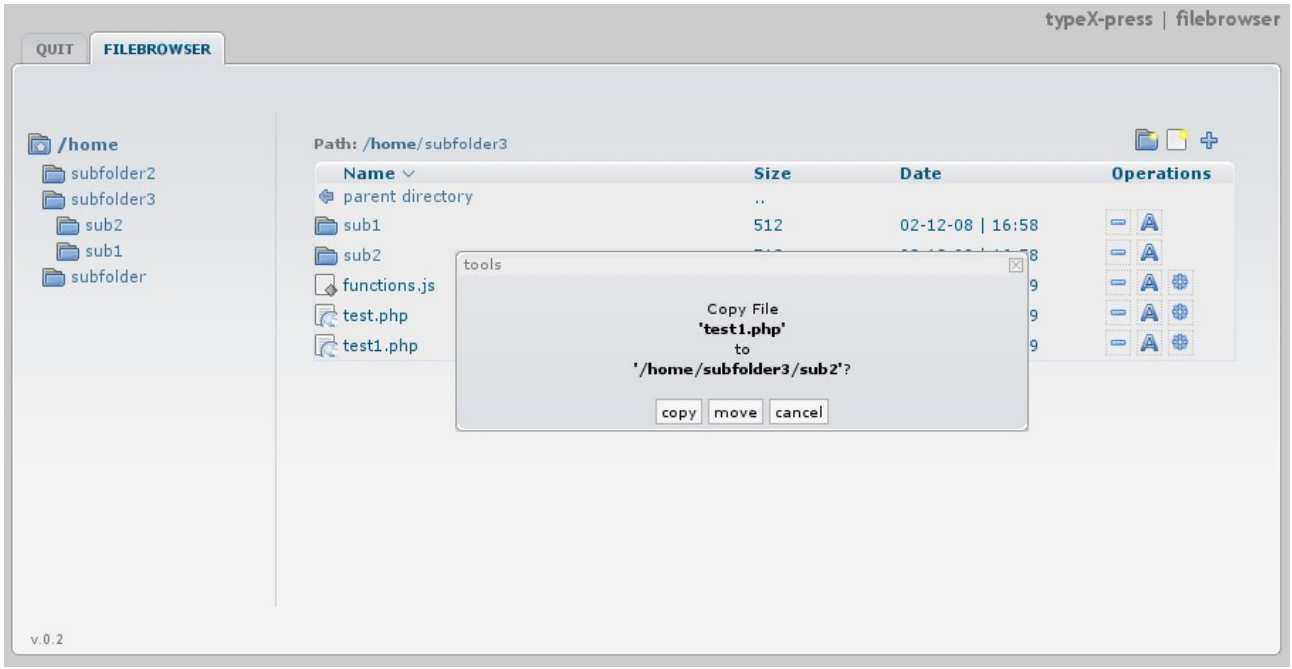

Abb. 11: typeX-press | Filebrowser

Der Dateimanager erlaubt es, Daten nach Größe, Datum und Name zu sortieren, Dateien und Ordner umzubenennen, zu erstellen, zu löschen und per Ziehen und Ablegen zu verschieben oder zu kopieren.

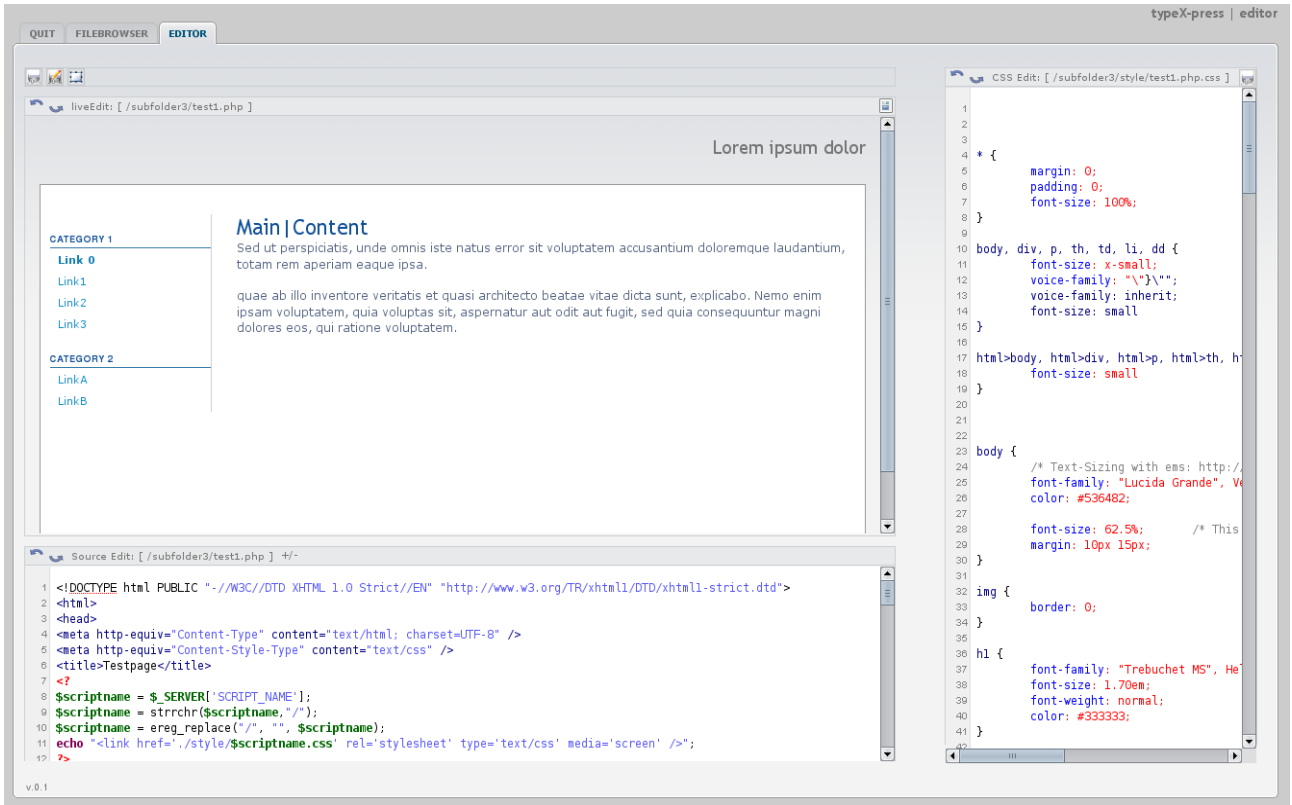

Abb. 12: typeX-press | Editor

Abbildung 12 zeigt übersichtsweise die Gliederung des Editors in ein Vorschaufenster, welches eventuelle Änderungen augenblicklich darstellt; den Quelltexteditor, den CSS Editor mit code-spezifischem Syntaxhighlighting sowie eine kleine Auswahl an Hilfsfunktionen.

6.2 Rahmenbedingungen des Projektes

Die Software lässt sich zu hundert Prozent auf freier Software einsetzen und ist plattformunabhängig. Lösungen die auf einen Webbrowsertyp beschränkt sind sollen vermieden werden und der innere Aufbau der Software möglichst modular und erweiterbar konzipiert sein. Eingesetzte Technologien sind PHP5, MYSQL, Apache, Subversion, und Javascript/Ajax (Abbildung 13). Der Quellcode der Software wurde unter der GPLv3 veröffentlicht und ist über das Internet erhältlich.

```
/////    AJAX request for the main browser table    /////

// Request senden // auf browser spezifische eigenschaften eingehen.. IE is mal wieder anders ^^
      function setRequest(dir,tsort) {
            if (window.XMLHttpRequest) {            // Request erzeugen
                  request = new XMLHttpRequest();   // Mozilla, Safari, Opera
            } else if (window.ActiveXObject) {
                  try {
                        request = new ActiveXObject('Msxml2.XMLHTTP'); // IE 5
                  } catch (e) {
                        try {
                              request = new ActiveXObject('Microsoft.XMLHTTP'); // IE 6
                        } catch (e) {}
                  }
            }

            if (!request) {                         // check request
                  alert("Kann keine XMLHTTP-Instanz erzeugen");
                  return false;
            } else {
                  var url = "./filebrowser_maintable.php";      //ziel url
                  request.open('post', url, true);              // open request
                  request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  // Requesthead
```

Abb. 13: typeX-press | Ajax-request

Nachstehender Screenshot (Abbildung 14) zeigt die erste Seite der Installationshilfe, welche die Einrichtung der Software, die Rechtevergabe und den Aufbau der MySQL Datenbank übernimmt.

Abb. 14: typeX-press | Setup

6.3 Open Source Aspekte von typeX-press

Plattformen wie „Sourceforge“ oder „BerliOS“ bieten Einsteigern der Softwareentwicklung alle Werkzeuge, die für einen schnellen Start notwendig sind, Webspace, Kommunikationsplattform, Versionsverwaltung und auch Hilfe bei der Wahl der freien Lizenz. Allein auf http://sourceforge.net sind mehr als 160000 Open Source Projekte angemeldet; eines von diesen trägt den Projektnamen „typeX-press“. Eine Mehrheit der Projekte entstand auf Grund persönlicher Bedürfnisse nach einer adäquaten Softwarelösung für eine Problemstellung und wurde aus nicht kommerziellen Beweggründen ins Leben gerufen. Da zumeist die persönlichen Ressourcen nicht ausreichen, um ein Projekt zu einem hohen Entwicklungsstand zu bringen, werden über jene Open Source Plattformen Ressourcen geteilt und Communities aufgebaut.

Im Falle meines eigenen Projektes ist die Tatsache, dass zum Entstehungszeitpunkt von „typeX-press“ bereits eine zweite, unabhängig entstandene Lösung für einen Online-Quelltexteditor bereit stand, die ein Kollege von mir entwickelte, hervorzuheben. Andreas Braun arbeitete zeitgleich an einer ähnlichen Software namens „editIT“.

Die Idee beider Projekte war es, zunächst eine einfache Möglichkeit zu schaffen, Inhalte online verändern zu können. So entstand sehr schnell ein reger Austausch von Codefragmenten, woraus die Idee entstand, die Codebasis beider Projekte zu verschmelzen und die Softwareentwicklung als Gemeinschaftsprojekt weiterzuführen. Die Zusammenlegung der Ressourcen war daher ein logischer Schritt, um gemeinsam schneller zum Ziel zu kommen und die Notwendigkeit eines Versionsverwaltungssystems kam auf. Aufgrund des hohen Bekanntheitsgrades, der Verfügbarkeit eines graphischen Userinterfaces (UI) für die eingesetzte Desktopumgebung KDE (Abbildung 15) als auch wegen der technischen Überlegenheit gegenüber anderen Versionsverwaltungssystemen wie CVS fiel die Wahl sehr schnell auf Subversion. Freies Zeitmanagement sowie eine hervorragende Ausrüstung in technischer Hinsicht ermöglichten darüber hinaus große Fortschritte in kurzer Zeit.

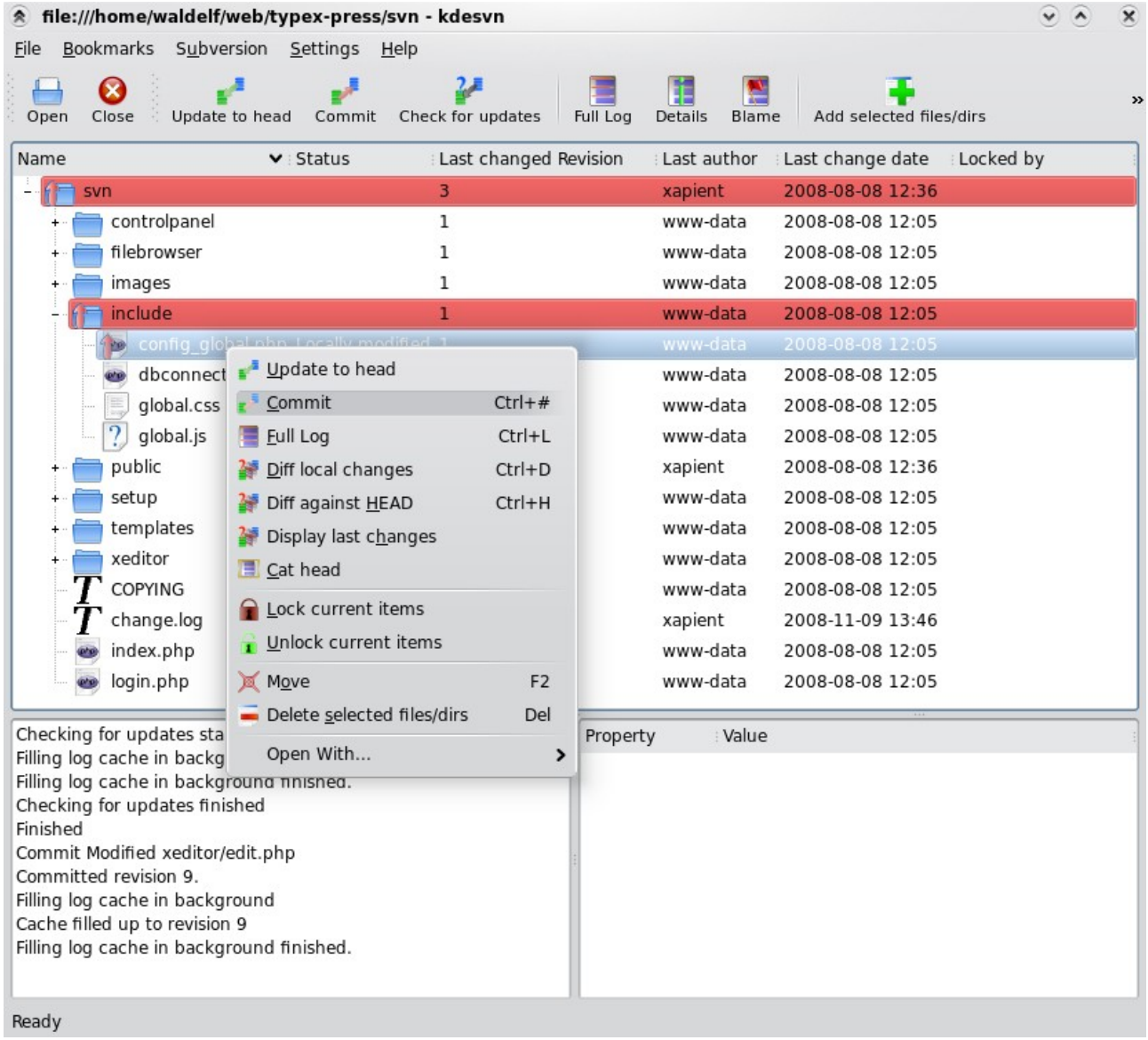

Abb. 15: KDE SVN | Subverison GUI

Die Lizenzierung unter der GPL und die Anmeldung des Projektes auf sourceforge.net, steht auch einer kommerziellen Nutzung nicht im Wege, da sowohl für die Programmierarbeit als auch für die Einrichtung der Software jederzeit eine finanzielle Vergütung verlangt werden darf. Die freie Lizenz ermöglichte es zudem auch, auf andere freie Systeme zurückzugreifen und somit „Codepress“, ein Skript, welches das Syntaxhighlighting im Editor übernimmt, sowie das „Tango“ Iconset zu verwenden. Dadurch konnte eine potentielle Mehrarbeit erspart werden und wir mussten das Rad nicht erneut erfinden, wie es in proprietärer Softwareentwicklung oftmals üblich ist.

Als Kommunikationsplattform, um Neuerungen zu besprechen bzw. zu veröffentlichen, installierte ich die Forensoftware „phpBB“. Da dieses Projekt noch in der Anfangsphase steht, dient die öffentliche Projektseite auf sourceforge.net lediglich der Information über die Software und der Weiterleitung auf die Seite *http://typexpress.xapient.net*. Damit ein Projekt wie dieses fortbestehen kann, bedarf es, wie in dieser Arbeit dargelegt wird, einer Community, die aufzubauen einer der nun folgenden Schritte sein muss. Die Einbindung mehrerer Benutzer und Tester sowie Programmierer muss ins Zentrum gerückt werden bzw. muss im Falle des Verlustes an Interesse an dem Projekt die Leitung desselben an einen engagierten Nachfolger abgegeben werden, was nach Eric Steven Raymond als letzte ehrenvolle Aufgabe in jedem Fall wahrgenommen werden sollte und zum guten Ton in der freien Softwareszene gehört. Momentan sind auf der Kommunikationsplattform fünf Benutzer registriert, die mit ihren Ideen am Projekt teilhaben, von welchen sich drei aktiv an der Entwicklung des Codes beteiligen.

6.4 typeX-press in der Schule

Auf Grund der inhaltlichen Ausrichtung des Projektes eignet sich dieses hervorragend zur Verwaltung einer Schulwebseite. Benutzer von typeX-press erhalten Schreibrechte auf einzelne Verzeichnisse der Webseite und können in diesen, ohne über Office-Management hinausgehende IT-Kenntnisse zu besitzen, eigene Inhalte veröffentlichen und bearbeiten. Interessierte Kustoden z.B. könnten fachspezifische Informationen unabhängig vom Zeitmanagement des Webmasters selbst aktuell halten. Ein prinzipieller Vorteil der Lizenzierung der Software besteht in dem Umstand, dass jedwede Veränderung und Erweiterung, die eine Schule an dieser vornimmt, der Community und somit allen anderen Schulen, die diese Software verwenden, zur Verfügung gestellt werden kann, ohne dabei in Gefahr zu laufen, eine Rechtsüberschreitung zu begehen. Eine bessere Vernetzung und Zusammenarbeit von Schulen bei derartigen Projekten und die Einbindung von typeX-press in den Informatikunterricht ist daher ebenfalls denkbar. Der modulare Aufbau, der die Erweiterung der Software einfach gestalten soll, gestattet es, Fotogalerien, automatisch erstellte Sprechstundenlisten, News-ticker und andere Module gemeinsam mit Schülern zu erstellen und PHP, MySQL, HTML und Javascript anhand dieser zu lehren und zu demonstrieren. typeX-press gestaltet sich somit nicht nur als typischer Vertreter von freien Softwareprojekten, sondern auch als Vorzeigebeispiel für deren Einsatz in allgemeinbildenden höheren Schulen.

7 Resümee

7.1 Auswertung der Interviews

Im folgenden Abschnitt versuche ich die Gemeinsamkeiten und Unterschiede der einzelnen Interviews hervorzuheben und eine grobe Übersicht über die gewonnenen Erkenntnisse zu geben. Ich versuche dabei, einzelne Aussagen auf ihren Sinn zu reduzieren, um dieser Zusammenfassung ihre Prägnanz zu erhalten.

Die persönliche Befragung praxisnaher Personen hat in vielen Bereichen eine signifikante Übereinstimmung der Antworten im Einklang mit den in dieser Arbeit berücksichtigten Studien und Thesen ergeben. So dürfte der verringerte Wartungsaufwand eines Linuxsystems dank dessen technischen Rahmenbedingungen und Möglichkeiten nicht nur theoretisch im Vordergrund stehen, sondern sich auch in der Praxis wiederholt bestätigen. Im Gegenzug postulierte jeder meiner Interviewpartner Praxiserfahrungen mit Windowsnetzwerken und begründete den wartungstechnischen Mehraufwand. Auch bei den Schwierigkeiten, die eine Migration bedeuten würde, herrschte Übereinstimmung, und Faktoren wie Gewohnheit und Unwissenheit standen an erster Stelle, noch vor allen technischen, die sich zumeist auf einen Mangel an alternativer Spezialsoftware zurückführen lassen. Letztere Schwierigkeiten sind freilich nicht außer Acht zu lassen und so finden sich auch in sehr homogenen Open Source Software-Landschaften vereinzelte Windowssysteme bzw. wie im BG Rechte Kremszeile unter WINE betriebene Programme. Auch gaben die befragten Personen an, zumeist auf Linux und ähnlicher Software zu arbeiten, in Einzelfällen aber auch auf proprietäre Produkte zurückzugreifen bzw. ein Dualbootsystem zu betreiben. Eine wesentliche Ursache für die flächendeckende Verbreitung proprietärer Software und dessen Beibehaltung trotz akzeptabler Alternativen scheint den Interviews zufolge in früheren Entscheidungen zu liegen. Lizenzkäufe des Ministeriums in der Vergangenheit haben zumeist dafür gesorgt, eine homogene Windowssystemlandschaft aufzubauen, welche durch diverse Folgekäufe gefestigt und etablierter wurde. Zu einer Zeit, in der Microsoft-Produkte die einzig sinnvolle Wahl darstellten, wurde der Grundstein für heutige Migrationsschwierigkeiten gelegt. Zur Zeit werden diese Verträge dank eines hervorragenden Lobbyismus seitens Microsoft vom Ministerium verlängert und Schulen somit diese Entscheidungen abgenommen.

Der sogenannte „Lock-In Effekt“ wird somit von allen Interviewpartnern als Hauptgrund für das zögernde Migrationsverhalten von Schulen genannt. Bei dem durchaus debattierbaren Standpunkt, Software respektive dessen Sourcecode wie Wissen zu behandeln, war man sich ebenfalls einig und postulierte die hohe Relevanz frei zugänglicher Ressourcen wie Software und dessen Quellcode für Bereiche, in denen eine gewisse Abhängigkeit von dieser Software vorhanden sei, jedoch sprachen sich zwei meiner Interviewpartner dafür aus, keinen generellen Standpunkt zu dieser Frage einzunehmen und nicht darauf zu drängen, dass der Quellcode einer jeden Software immer frei sein muss. Auch ob Quellcode an sich Wissen beinhalten würde, wurde in Frage gestellt. Wissen über objektorientierte Programmierung und ähnliches würde man sicherlich nicht aus dem Quellcode von Software erlangen, sondern eher über einschlägige Bücher. Das Wissen über die spezifische Lösung eines Problems und die Art der Implementierung diverser Features mag jedoch aus diesem abzulesen sein, was u.a. den Vorteil der gemeinschaftlichen Problemlösung und Verbesserung desselben bedeuten kann. Die ethischen Hintergründe einer Entscheidung für oder gegen freie Software wurden von allen Befragten als wertvoll erachtet und zu einem gewissen Teil in den Verantwortungsbereich von Schulen gelegt, deren Aufgabe es u.a. auch ist, Persönlichkeitsbildung und gesellschaftliche Entwicklung zu fördern. Praktischer Nutzen und finanzielle Überlegungen wurden jedoch generell vorgezogen und in das Zentrum der Betrachtung gerückt. Auch die Offenheit des Quellcodes wurde durchwegs im Vordergrund praktischer Überlegungen behandelt und Systemsicherheit und schnelle, evolutionäre Entwicklung als Vorteile genannt, aber auch das erhöhte Risiko durch den offen liegenden Code und somit auch eventuelle Sicherheitsmängel erwähnt sowie die denkbare Möglichkeit, veränderte, schädliche Softwarepakete als vermeintliches Original in Umlauf zu bringen, was dem bereits gängigen „Fishing“ bei Websites entsprechen würde. Von der Möglichkeit, den Quellcode eines Programms im schulischen Rahmen zu verändern, Gebrauch zu machen, scheint hingegen eher unvorstellbar. Zu tief müsste man in die Materie eintauchen, was den Zeitrahmen von Lehrern und Schülern sprengen würde. Über eine obligatorische Anpassung des Systems bzw. des Bootvorganges hinaus an eigene Bedürfnisse, welche durch den modularen Aufbau, die Offenheit des Systems, die Lesbarkeit der Konfigurationsdateien und die ausführliche Dokumentation gewährleistet wird, werden keine tiefer gehenden Veränderungen am Code der Anwendersoftware vorgenommen.

Die Interviewpartner sind sich einig, dass man den echten Code im Unterricht selbst nie bearbeiten wird und diesen höchstens zu Demonstrationszwecken vorführen könnte. Die Offenheit von Schnittstellen und die Notwendigkeit von offenen Standards hingegen wurde generell nicht nur aus verwaltungstechnischen Gründen, sondern auch aus einer Lehrsicht heraus als wichtig erachtet. Kompatibilität beim Austausch von Daten steht in diesem Fall im Vordergrund und somit nicht gezwungenermaßen offene Standards, wenngleich letztere diese Ansprüche eher erfüllen können, ohne eine Quasi-Monopolstellung der Anwendungssoftware, die auf diese zurückgreift, vorauszusetzen. Diese Anwendersoftware sollte im Unterricht einer AHS austauschbar und plattformübergreifend einsetzbar sein. Auch wenn die Auswertung der Interviews ergeben hat, dass es nicht die Aufgabe einer Schule sein kann, wettbewerbsverzerrenden Verhältnissen in einer globalen Marktwirtschaft entgegen zu wirken und auch in Frage gestellt wird, ob dies überhaupt möglich ist, so liegt es dennoch im Zuständigkeitsbereich einer Schule, solche Entscheidungen unter Rücksicht auf alle Beteiligten zu treffen und auch bei finanziellen Überlegungen den Aufwand seitens der Schüler und Eltern zu bedenken. Dies wiederum führt zu einer indirekten Einflussnahme auf diesen Umstand, da frei kopierbare, plattformübergreifende, zumeist freie Software sich als vorteilhafte Lösung aufdrängt, das Problem der illegalen Vervielfältigung eliminiert und die im schulischen Kontext eingesetzte Software mit Sicherheit Einfluss auf später präferierte Softwarelösungen hat, selbst wenn im Unterricht speziell darauf geachtet wird, Konzepte - nicht Programme - zu lehren. Auch in diesem Punkt findet sich in allen Interviews die kohärente Ansicht, dass Methoden und Abläufe, technisches Grundverständnis und Grundbegriffe gelehrt werden sollten anstelle von spezieller Produktschulung und dass Informatiklehrer die Aufgabe haben, hier mit Weitblick vorurteilsfrei zu agieren und eigenständiges Denken anstelle von sturem „hinklicken“ zu fördern.

7.2 Zusammenfassung

Benutzen, verändern, verschenken, nachahmen, etc. ist gestattet. Freie Software gewährt dem Benutzer all jene Freiheiten die er auch bei anderen Konsumgütern erfährt.

Freie Software wird all zu oft in Verbindung mit „gratis“ gebracht. Es heißt: „Etwas das umsonst zu bekommen ist, taugt nichts“. Die Realität sieht jedoch ganz anders aus. Schon 2007 hatte der Mozilla Webbrowser „Firefox“ einen Marktanteil von über 30% und vermutlich viele Fans in den Reihen der Benutzer, die über die Freiheit der Software gar nicht Bescheid wussten. Ein Blick hinaus in die Welt zeigt: Der ideologische Hintergrund der „Free Software Szene“ scheint oft nur für Programmierer, Informatiker und Idealisten ausschlaggebend zu sein, auf FOSS zu setzen. Auf Grund von Werten wie Wissensfreiheit, einer antikapitalistischen Haltung, dem Wunsch wirtschaftlich unabhängig zu sein, einer großen Community anzugehören, etwas an die Gesellschaft zurückzugeben, etc. wird von diesen oft der Einsatz von freier Software von ihnen propagiert und forciert.

Im Gegensatz dazu stehen für viele User und öffentliche Einrichtungen zumeist eine Reduzierung der wirtschaftlichen Abhängigkeit, der damit verbundene finanzielle Vorteil und die technischen Hintergründe im Vordergrund. „Upgrade-Zwang“ von Software und Hardware, Kompatibilitätsprobleme zwischen den Versionsnummern, Unterschiede zwischen privater und beruflicher (schulischer) Softwareausstattung etc. sind Probleme, mit denen Schulen heute zu arbeiten haben. Der Umstieg auf freie Software schafft hier in der Regel eine Harmonisierung.

Zusätzlich wird die Tatsache, dass der Quellcode des eingesetzten Programms zur Verfügung steht und somit öffentliche Einsichtnahme ermöglicht, oft als relevanter Vorteil von FOSS angeführt. Da es kein „Non Disclosure Agreement“ wie bei proprietärer Software gibt, können bei Sicherheitsprüfungen gefundene Fehler veröffentlicht und die Anwender schnell informiert werden, um Gegenmaßnahmen zu ergreifen. Für viele ist die Wahl der Softwarelizenz ist gleichzusetzen mit der Entscheidung „Vertrauen versus Wissen“ [vgl. *BSI*]

Durch das Internet sind sowohl die Produktions- als auch die Vertriebsmittel, Computer und Netzzugang, nicht mehr in der Hand einiger weniger; so kann jeder mit dem notwendigen Know-how zu einem Softwareentwickler werden und auch die rasche Verteilung der Software ist gewährleistet. Programmierer und Supporter können sich durch Schaffen und Unterstützen von „freien Software Projekten“ selbst verwirklichen und das Endprodukt nach ihren Vorstellungen gestalten. Das vorherrschende proprietäre Modell erlaubt den Anwendern keine Einflussnahme auf die Anwendungssoftware bzw. lässt sie in der Abhängigkeit des guten Willens der Hersteller verweilen. Ein Zustand, der für öffentliche Einrichtungen oft nicht tragbar ist. Zudem erwirkt freie Software durch das etablierte Entstehungsmodell - mit Unterstützung einer Community - Konkurrenz zu etablierten Quasi-Monopolisten, wo der Versuch, dies durch kapitalistische Produktionssysteme zu erreichen, u.a. wegen Unfinanzierbarkeit oft gar nicht erst in Erwägung gezogen wird. Eine klare Stärke von freier, quelloffener Software ist jene, dass es auch in ferner Zukunft eine Möglichkeit geben wird, freien Systemen wie Linux jeden erdenklichen Vorteil abzuverlangen. So können diverse Systeme darauf basieren, es können Linux und dessen ausführliche Dokumentation als “Blueprint” herangezogen und nachgelesen werden, weshalb eine bestimmte Lösung für ein Problem genutzt wurde, weshalb der Code auf jene Weise entworfen wurde und vieles mehr.

Für allgemeinbildende höhere Schulen, die in der Regel nur wenig Spezialsoftware einsetzen, die durch das Angebot freier Software nicht gedeckt werden könnte und freie Software zumeist auch keine aktuelle Hardware benötigt, gilt nach Ansicht diverser Studien, dass ein gut geführter Umstieg einen finanzielle Besserstellung bedeuten kann. Der Wartungsaufwand kann langfristig reduziert und der Sicherheitsstandard erhöht werden. In Supportfragen kann auf die Community zurückgegriffen werden bzw. dieser Support auf herkömmliche Weise gekauft werden. Durch den Einsatz von freier Software besteht für die Schulen erstmals die Möglichkeit, sich zumindest teilweise aus der strengen Abhängigkeit einzelner Softwareanbieter zu lösen. Die Unterstützung offener Formate stellt sicher, dass die Daten auf lange Sicht hin lesbar und weiter verarbeitbar bleiben.

Das Ressourcen schonende Design von GNU/Linux ermöglicht, auch ältere Hardware sinnvoll einzusetzen und auf diese Weise die Anzahl der nutzbaren Personal Computer zu erhöhen. Zudem entfallen für Schulen, Schüler und Lehrer die Lizenzkosten, was sich positiv auf die Kosten der IT auswirkt. Die durch den Einsatz freier Software verringerten Kosten für schulinterne IT können durch Rückführung der finanziellen Mittel in den Unterricht respektive in Unterrichtsmaterialien infolgedessen zu einer Steigerung der Qualität des Lehrangebotes führen. Eine wesentliche Hürde, die es zu überwinden gibt, ist der Lock-in der Belegschaft sowie die einfache Gewohnheit der einzelnen. Vorausgehende Informationsveranstaltungen und Schulungen sind daher eine sinnvolle Ergänzung und Grundlage einer erfolgreichen Migration. Eine Migration bedeutet immer eine Umstellung aller Beteiligten auf eine neue Situation und ist daher mit einem angemessenen Zeitrahmen zu planen. Dieser Umstellung setzt eine gewisse Vorlaufzeit voraus, in welcher das neue System vorgestellt werden kann, bevor es zum praktischen Einsatz kommt. Für einen zukünftigen reibungslosen Umstieg auf freie Software ist vor allem bei Ausschreibungen darauf zu achten, nur Software zu erwerben, die den heutigen Anforderungen entsprechen und plattformunabhängig betrieben werden kann. Dies wäre eine empfehlenswerte Herangehensweise, die vor allem auch vom Ministerium unterstützt werden sollte, das im derzeitigen Modell die Lizenzen für die Schulen erwirbt. Langfristig kann hier also nur ein Umdenken zu einer Veränderung der Situation führen, das auch in hohen Kreisen um sich greift und den Schulen bei der Wahl ihrer Software wieder mehr autonomes Entscheiden, angepasst an den Schultyp und den individuellen Bedingungen, gewährt. Die Aufarbeitung des Themas dieser Arbeit im Kapitel „FOSS im Schulbetrieb“ hat ergeben, dass freie Software eine ausgesprochen gute Alternative zu kommerziellen, kostenpflichtigen Produkten darstellen kann, sich ausgezeichnet für den Einsatz im Unterricht eignet und den Wartungsaufwand langfristig senken kann. In gewisser Weise gehört es zur sozialen Verantwortung einer allgemeinbildenden höheren Schule, diese zumindest in Betracht zu ziehen und für ihre spezielle Situation zu prüfen, wie und in welchen Bereichen sie am besten zum Einsatz kommen kann. Anders zeigt sich die Situation, wenn höhere technische Lehranstalten und andere berufsbildende Schulen auf Grund ihrer Zielsetzung tatsächlich Produktschulung betreiben bzw. wenn sie auf Grund der Hardwareausstattung und aus Mangel an Alternativen gezwungen sind, auf proprietäre Systeme zurückzugreifen.

Nicht Einsparungen an Softwarelizenzen sollten das alleinige Ziel sein, sondern in der Folge die Verlagerung der Ausgaben von Software zu Hardware und somit eine bessere Ausstattung für Schule und Unterricht. Formale Richtlinien zur Bewertung und Kategorisierung von Open Source Nutzung müssen landesweit entwickelt werden und Schulen auf eine kommende Verlagerung der Wertigkeit im Softwarebereich vorbereitet werden. Die Studie der Materie hat ergeben: Free Open Source Software wird in Zukunft ein Faktor sein, mit dem man zu rechnen hat.

7.3 Ausblick

> *„Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software ``hoarding'' is morally wrong (assuming you believe the latter, which neither Linus nor I do), but simply because the closed-source world cannot win an evolutionary arms race with open-source communities that can put orders of magnitude more skilled time into a problem.“[TCAB, S.23]*

Projekte wie „One Laptop Per Child“ (OLPC) und das „OpenMoKo“ ebnen schon jetzt den Weg in eine Zukunft, in welcher freie Software in alle Lebensbereiche Einzug halten wird. Kleine Notebooks oder auch Netbooks verlangen nach einem modularen, hoch-skalierbaren System wie dem Linuxkernel. Die freie Verfügbarkeit der Software sowie die Bereitschaft der Communities, einen Beitrag zu leisten, machen auch für Firmen den Einsatz von Linux attraktiv. Am Beispiel des OpenMoKo Projektes, welches mit dem GTA01 die erste, vollkommen freie Hardware präsentierte und inzwischen an der dritten Version derselben arbeitet, lässt sich der Einzug von Linux auch auf „ultramobile“ Devices hervorragend beobachten. QTExtended bzw. Android, beides ebenfalls Linux-basierende Betriebssysteme für Handys, fügen sich in die Reihe freier Software für mobile Devices ein. Für weniger technikaffine Endbenutzer nicht bemerkbar wird auf einer wachsenden Anzahl kleinerer Geräte bereits mit freier Software gearbeitet und Netzwerkrouter mit Linux als Betriebssystem stehen schon heute in unzähligen Haushalten mit Internetanschluss.

Der Einzug in verschiedenste Lebensbereiche und die steigende Akzeptanz freier Software wird auch vor Schulen nicht halt machen und die höhere Notwendigkeit technischer Hilfsmittel lässt die Frage nach der Lizenz einer Software wichtiger erscheinen. Es gilt, das „Wir“ einer Schule zu erweitern und Eltern, Schüler und Lehrpersonal in einen Entscheidungsprozess mit einzubinden, der vor allem die Eltern der Schüler auch finanziell betrifft. Schwarzinger zufolge wird auf alle Beteiligten einer Schule, durch die Wahl der Software(-lizenz) indirekt Einfluss genommen. Da die Verwendung identer Softwareprodukte sowohl innerhalb als auch außerhalb der Schulen sinnvoll erscheint, wird man indirekt dazu veranlasst, sich die in der Schule eingesetzte Software auch für den Heimcomputer anzuschaffen, was im Falle proprietärer Software einen unnötigen, finanziellen Mehraufwand für Eltern bedeutet. Mit zunehmender Wichtigkeit von Software im Schuleinsatz steigt daher auch die Relevanz kostengünstiger, wartungsarmer IT Lösungen, was dazu führt, dass *freie Software* im Zentrum zukünftiger Entscheidungsprozesse stehen wird.

8 Anhang

8.1 Interviewleitfaden

Die Free Software Foundation (FSF) definiert freie Software durch 4 Freiheiten:

1.Die Freiheit, das Programm für jeden Zweck auszuführen.
2.Die Freiheit, die Funktionsweise eines Programms zu untersuchen und es an eigene Bedürfnisse anzupassen.
3.Die Freiheit, Kopien weiterzugeben und damit seinen Mitmenschen zu helfen
4.Die Freiheit, ein Programm zu verbessern und die Verbesserungen an die Öffentlichkeit weiterzugeben, sodass die gesamte Gesellschaft profitiert.

- **1) Setzen Sie freie Software privat ein?**

- **2) Benötigen Sie Anwendungssoftware, die nicht oder nur unzureichend durch das Angebot freier Software gedeckt würde?**

Viele Menschen vertreten die Meinung, dass Software respektive der Quellcode der Software zu handhaben sei wie Wissen, welches jedem zugänglich sein sollte, dessen Weitergabe das Recht und durchaus auch die Pflicht eines jeden sozialen Wesens ist.

- **3) Was halten Sie von diesem Statement?**

Durch den ideologischen Background wird die Wahl der Software eine philosophische und ethische Wahl, die es zu treffen gilt, und auch eine politische.

- **4) Sollten und können Überlegungen dieser Art bei der Wahl der Software für eine Schule von Bedeutung sein?**

- **5) Offener oder geschlossener Quellcode? Welche Bedeutung hat dies Ihrer Meinung nach für die Sicherheit einer Software?**

- **6) Warum, glauben Sie, wird in Schulen nach wie vor in so hohem Maße auf proprietäre Software gesetzt?**

- **7) Was spricht ihrer Ansicht nach für einen Umstieg auf freie Software?**

- **8) Mit welchen Schwierigkeiten hat man bei einer Migration zu rechnen? (**Verwaltung, Lehrende oder Lernende)

Die FSF definiert als 2. Freiheit, die „freie Software“erfüllen muss, die Freiheit, die Funktionsweise eines Programms zu untersuchen und es an individuelle Bedürfnisse anzupassen.

- **9) Ist die Möglichkeit, die Funktionsweise eines Programms zu verändern, relevant für Bildungseinrichtungen?** Informatikunterricht? Warum? Auf welche Weise?

Offene Schnittstellen und die damit entstehende Interoperabilität sind ein wesentliches Fundament von Free Open Source Software.

- **10) Könnten Sie in wenigen Worten erläutern, warum offene Schnittstellen und Standards von Bedeutung sind?** (in inwiefern ist das für eine Schule wichtig?)

- **11) Ist die illegale Vervielfältigung von Software ein Problem, mit dem sich Schulen auseinandersetzen müssen?**

Die Wiederverwertbarkeit bzw. das Recht zur Vervielfältigung ist Inhalt der dritten Grundsatzdefinition freier Software.

- **12) Wie sollte, im schulischen Kontext, die Software-spezifische Ausbildung junger Menschen gestaltet sein?**

Microsoft hat mit seiner Office Suite und dem Betriebssystem Windows ein sogenanntes Quasi-Monopol in der Softwarelandschaft.

- **13) Kann es die Aufgabe einer Schule sein, wirtschaftlichen Monopolen entgegen zu wirken?**

- **14) Gibt es etwas zu diesem Thema, das wir noch nicht besprochen haben, aber noch gesagt werden sollte?**

- **Danke!**

8.2 Transkribierte Interviews

8.2.1 Mag. Florian Wisser

Steckbrief:
Studium der Mathematik
Softwareentwicklung in Java
Unterricht an einer Fachhochschule
Unterricht und Systembetreuung an der Universität Wien
Linux Lehrgänge für IBM.

1) Setzen Sie freie Software privat ein?

Ja, ich bin seit 10 Jahren Linuxbenutzer, auch am Desktop. Zuhause arbeite ich seitdem außer für Einzelanwendungen in der Tonbearbeitung, weil es hier die Standardtools auf Windows gibt; aber das verwende ich selten. Also wenn ich meinen Rechner daheim aufdrehe, dann arbeite ich zu 98 Prozent mit Linux. Ich hab von Windows 3.1 auf Linux gewechselt, weil ich Java programmieren gelernt habe. Win3.1 war noch ein single task System und da hatte ich die Wahl, Windows95 auszuprobieren, was damals sehr schlechte Kritiken gehabt hat, daher hab ich mir gedacht, ich probiere es aus und das war dann Redhat 5.2. Das ist damals auf einer CD gekommen und ich hab die Installations-CD eingelegt, bin auf die Uni studieren gefahren und wie ich heim gekommen bin, war die Installation fertig. Das war im Jahr 95-96, das weiß ich nicht mehr so genau, wann die wirklich released haben, aber so was in der Größenordnung.

2) Benötigen Sie Anwendungssoftware, die nicht oder nur unzureichend durch das Angebot freier Software gedeckt würde?

[...] Also es gibt relativ wenig. Die Leute, die bei uns programmieren, sind eher auf Linux zuhause. Ein Kollege hat, glaube ich, noch einen Cluster, den er unter Windows betreibt und mit Fortran programmiert, aber das ist eher seinem Alter zuzuschreiben. Was sonst noch wichtig wäre; am Server - das ist ganz interessant - haben wir jetzt eine relativ große Serverinfrastruktur gekriegt. Früher, seit einem Jahrzehnt haben wir praktisch nur Linux server benutzt, dann viel Geld investiert und ein heterogenes Feld aus Sunservern mit SunOS oder Solaris und Linux bekommen. Das Solaris war nicht dazu zu bewegen, in der Standard Installation eine Wiki mit unter 5 Sekunden Antwortzeiten über den Webserver auszuliefern. Unsere Administratoren und auch ich haben es probiert.
(Was läuft für ein Webserver?) Apache - ganz einfach - ich wollt den Rechner nur benutzen und er war unbenutzbar in dem Zustand. Unsere Administratoren haben sich über mehrere Monate zumindest versucht einzulesen und haben es nicht in einen Zustand gebracht, der auch nur einigermaßen befriedigend gewesen wäre, während die Standard Linux Installation auf einem ganz normalen Intelrechner, das war sogar ein "Celeron-irgendwas", also wirklich ein billiges Gerät, es problemlos erledigt hat. Ich hab mich auch über die Sun wahnsinnig geärgert, weil ich auch eine aktuelle Maschine gehabt habe, die hätte zwar sehr schöne Virtualisierung gehabt - das ist wirklich super gelöst - aber nur um das Ding mal zu benutzen, das heißt, was nach zu installieren, ist so mühsam. Die Virtualisierung war einer der Beweggründe.. jetzt haben sie eh XEN. Mailserver ist mittlerweile draußen, unsere Website läuft jetzt unter Plone und dahinter ein Zope, wie das halt so üblich ist, das kann man debattieren. Sonst ist es jetzt so.. es ist ein recht heterogenes Umfeld, denn wir haben einen Windows Domaincontroller, weil du die Windows Workstation schwer ohne einen Domaincontroller betreiben kannst, du könntest schon aber, naja, Profile wohin spielen, etc. Profilverwaltung, nicht für die Studenten, für die Angestellten. Im Pc-Labor gibt es auch dualboot, da haben die Studenten aber ein unveränderbares Minimalprofil, das ist Server diktiert und um das so zu machen, ist es unter Windows am einfachsten, einen Windowsserver hinzustellen. Dann gibt es diese Policyschalter, und man kann sagen: der sieht jetzt seine Dosbox nicht, usw.; es war zwar nicht ganz trivial und zum Teil auch ein wenig idiotisch, aber die Einschränkungsmöglichkeiten waren gut. Ja, und das LDAP wird auch von den Linuxclients benutzt, active directory wird zur Anmeldung auch benutzt.

Mathematica bei uns ist sicher der Hauptpunkt dabei, was bei uns wirklich viel verwendet wird, ist Mathematica und Mathlab, die beide nicht frei sind. Maple wird nur am Rande benutzt, das kann man eigentlich vergessen, das wird sowohl in der Forschung als auch in der Lehre benutzt und „nicht abdecken“ ist immer so eine Frage. Also z.B. Mathlab ist für die Leute, die numerisch rechnen, praktisch einfach DAS Ding, die Referenz in gewisser Weise, oder wenn du z.B. im wissenschaftlichen Bereich numerische Kalkulationen hast und die Referenz, bzw. alles was in Referenz programmiert ist, ist in Mathlab programmiert, dann tust du dir auch schwer, von dem weg zukommen. Oktave hat das geheißen, das war so ein Mathlabclone, das hat versucht, das Mathlabinterface zu imitieren, aber mir hat dann wer gesagt, es hat dann Programme gegeben, die behauptet haben, sie laufen auf beidem, die sind dann im Prinzip auf große „if Mathlab then .. elseif Oktave“-Blöcke hinausgelaufen, was natürlich vollkommen vertrottelt ist. Mathematica hat zum Beispiel in der Lehre sehr große Vorteile, weil es sehr einfach ist. Gerade Mathematica 6 erlaubt es, so bunte bewegte Bilder zu machen, es ist sehr hübsch gemacht und es gibt jetzt auch eine Veranstaltung, die heißt „Die Hilfsmittel aus der EDV“, die eine Einführung in Computeralgebra ist, wo man sich für eine Lösung entscheiden muss und wir haben lange überlegt. Auf der Uni Hannover macht Martin Rubey, der dort jetzt hingeht, das ganze mit „Seetch“, das ist ein Open Source Äquivalent, das noch nicht soweit war, als wir vor einem Jahr angefangen haben. Wir haben uns dann schweren Herzens aber doch für Mathematica entschieden, nicht zuletzt deshalb, weil der ZID das in die Hand genommen hat, damit Studierende Billiglizenzen bekommen. Wenn das zu Ende ist, schaut es wieder anders aus, weil das wirklich teuer ist. Mathematica kostet für ein Kalenderjahr 120 Euro für einen Studenten und jetzt bekommen sie es um 15. Man kann es in den Pc-Labors natürlich frei benutzen, es gibt natürlich auch andere Möglichkeiten, es zu bekommen, die natürlich nicht legal sind.

3) Was halten Sie von diesem Statement?

Ich würde sagen, teilweise beinhaltet es Wissen, es beinhaltet oft gar nicht so viel Wissen... schlecht programmiert oder wenn man einfach dumm an etwas herangeht.
Natürlich gibt es im Programmcode immer wieder intelligente Lösungen zu finden. Wenn allerdings jetzt jemand dieses Wissen im wissenschaftlichen Bereich wirklich hergeben will, dann ist es typischerweise so, dass er das in ein kleines Paper mit Pseudocode schreibt, in dem es für einen User einfacher zu lesen, also es für einen Leser einfacher ist, die Ideen herauszuziehen als direkt aus dem Quellcode.. da steht halt „ich verwende im Prinzip die Methoden von ... aber da hab ich mir etwas gedacht, daher ändere ich das ein bisschen ab.“ Ich denke auch, dass es die intelligentere Art der Wissensvermittlung ist, ob das jetzt über den Quellcode selber passiert oder über andere Quellen, das ist fraglich und ob man jetzt z.B. den ganzen Quellcode von was auch immer freigeben muss, das sehe ich nicht mehr so religiös, wie ich es vielleicht einmal gesehen habe. Ich kann nicht sagen, ich kämpfe dafür, dass alle Leute immer ihre Quellcodes hergeben, aber es ist sicher sehr praktisch in vieler Hinsicht und ich denke, man sieht daran z.B., dass Sun jetzt Java freigegeben hat, sodass die Leute im Prinzip in die Umgebung eingreifen könnten, auch von außen - auch wenn sich die Frage stellt, ob sie das aufnehmen - aber man sieht daran, dass wenn man einen community betriebenen Entwicklungsprozess haben will, dann führt kein Weg daran vorbei. Aber wenn es nur darum geht das Wissen zu vermitteln, dann kann man das, also wenn ich z.B lesen will, wie man „Alpha-beta pruning“ in einem Spiel-tree macht, dann werde ich nicht ein Spiel beginnen zu lesen bzw. den Sourcecode von einem Spiel, sondern ich nehme mir ein Buch her, wo drinnen steht, wie das geht. Ich glaube, tendenziell wird das viel zu wenig getan, allerdings stellt sich auch oft die Frage, wie sehr es wirklich praktisch ist, etwas her zu nehmen, das es schon gibt, was kein Library ist und nur das zu ändern, was zu ändern ist
(Die Idealform wäre in diesem Falle eine Art Library wiederzuverwenden, wie es in der OS Gemeinde üblich ist ?)

Ja natürlich, das ist das Ideal. Man muss das Rad nicht neu erfinden, es ist dann idealerweise auch noch extern dokumentiert. Das ist der Punkt wo ich sage; es muss nicht immer alles im Sourcecode vorhanden sein, weil sonst brauchtest du ja z.B., wenn „der“ die ultimative Referenz wäre, keine Dokumentation für das Library, weil das Library ist eh schon da, dann nimmst du es halt einfach und verwendest es.

Die Wissensvermittlung ist oft nicht direkt verbunden mit der Frage, ob man die Sourcen beigibt oder nicht.

4) Sollten und können Überlegungen dieser Art bei der Wahl der Software für eine Schule von Bedeutung sein?

Könnten sie prinzipiell, also ich finde, könnten sie schon, allerdings hast du das Problem, wenn ich mich jetzt z.B. auf den Standpunkt stelle, dass OS2 das ultimative Betriebssystem war in politisch korrekter Hinsicht, dann kann ich zwar die Idee haben, damit zu arbeiten, aber ich werde kläglich daran scheitern, weil es einfach technische Randbedingungen gibt und bevor nicht diese technischen Randbedingungen erfüllt sind, wie benutzbare Software für das und das und das, dann kannst es eh vergessen, dann kannst du die Politik wieder vergessen. Wenn also Dinge zur Auswahl stehen, die man beide betreiben kann, dann kannst du dir das überlegen, ob es die richtige Überlegung ist, ist fraglich. Was man natürlich sagen kann ist, man will z.B nicht so viel Geld für Lizenzen zahlen, das ist natürlich überhaupt ganz tendenziell ein Problemfeld, weil du, wenn du dir überlegst, dass du z.B. Schulungen für proprietäre Software machst, tust du da nicht quer subventionieren? Wir haben z.B ein Hilfsmittel in dieser EDV-Einführung „Einführung in Mathematica“. Indirekt sponserst du damit natürlich Wolfram aus staatlichen Geldern, weil du die Leute daran gewöhnst es zu benutzen und es später wahrscheinlich das Erste sein wird, was sie in die Hand nehmen. Andererseits ist es halt so, ich zeig ihnen zum Schluss dann, es gibt noch etwas anderes auf der Welt, andererseits ist alles andere in diesem Fall auch komplizierter in der Handhabung, also du verlierst einfach, wenn du es anders machst. Du hast 20% drin sitzen, die ohnehin alles können, die könntest auch damit allein lassen, 20% sind sowieso überfordert und werden das nie wieder anschauen und die in der Mitte verlierst du halt komplett, wenn es kompliziert wird, und ja, es ist in der Handhabung in gewisser Weise komplizierter.

5) Offener oder geschlossener Quellcode? Welche Bedeutung hat dies Ihrer Meinung nach für die Sicherheit einer Software?

Also da bin ich im Prinzip ganz auf der offenen Seite ... es gibt in der Kryptographie das „kerkhoffsche“ Prinzip, das sagt, ein Verschlüsselungsalgorithmus soll so konzipiert sein: Die Sicherheit darf nicht am Algorithmus hängen. Der muss publiziert sein. Die Sicherheit darf einzig und allein am Schlüssel hängen. Im Prinzip, wenn du nur nicht weißt, wie verschlüsselt wurde, ist das praktisch „security by obscurity“, weil die Sicherheit nicht in dem Schlüssel steckt. Bei allen vernünftigen Standards wie AES z.B. sieht es folgendermaßen aus: jeder Mensch kennt den Algorithmus, jeder Mensch kann ihn lesen, aber er kennt den jeweiligen Schlüssel nicht, also kann er es nicht entschlüsseln und so denke ich, ist es auch mit der Sicherheit von Software. Nur weil ich nicht weiß, wo der Fehler von einem Programm ist, heißt das nicht, dass ich ihn nicht angreifen kann. Also da bin ich sicher auf der Seite der Open Source, weil es einfach möglich ist, das zu sehen. Es bietet vielleicht auch mehr Angriffsflächen, die aber schneller kleiner werden. Ich denk mir eher, die größeren Probleme entstehen vielleicht durch andere Dinge. Es gibt z.B. so Analysetools, die Fehler finden. Jetzt könnte der Angreifer das Programm natürlich da hineinstecken und findet sicher in jeder Software etwas, ob er das aber dann ausnutzen kann oder nicht, ist dann einen andere Frage. Es kommt auch auf den Fehler an und die typischen Angriffe sind wahrscheinlich eher von niedriger Qualität, aber ich denke schon, dass dies über freie Software á la longue wesentlich besser funktioniert.

6) Warum glauben Sie wird in Schulen nach wie vor in so hohem Maße auf proprietäre Software gesetzt?

In der Schule hatte es zum Beispiel auch einmal die Computeralgebra Debatte gegeben und das Ministerium hat eine schulweite Lizenz für „Derive“ gehabt (mittlerweile aufgelassen), welches dann auch in „embedded“ Form auf dem TI (Texas Instruments) drauf war. Derive war ein reines Windows Programm und das bindet schon mal. Das war eine frühe Entscheidung, die haben halt zuerst auf irgendwas gearbeitet, das Ministerium hat dann geschaut, welche Computeralgebra Systeme gibt es, die das abdecken können, was bekommen wir am billigsten .. das war „Derive“.

Was anderes als Windows hat es damals nicht gegeben. Jetzt kaufen wir es für 5 Jahre. Nun ist das praktisch vorbei, aber ich glaube schon, dass es solche Prozesse sind, die einen dazu bringen, dabei zu bleiben. Du weißt, du bleibst bei Windows, weil du 5 Jahre lang „Derive“ hast, im Jahr 3 dieser Laufzeit brauchst du einen Domaincontroller, dann hast du einen Domaincontroller, dann ist es schwer wieder die anderen Teile zu wechseln und so geht das halt immer weiter... ja, um da raus zukommen, musst du dann Einschnitte machen, die im Prinzip den Usern total unangenehm sind, wenn du das aus einer administrativen Sicht siehst. In Schulen geht es wohl insofern ein bisschen leichter, weil du dann sagen kannst, ich starte jetzt einen neuen Jahrgang und der wird umgestellt, wobei du das natürlich nicht ganz unabhängig machen kannst, weil natürlich eine fixe Infrastruktur existiert.

7) Was spricht Ihrer Ansicht nach <u>für</u> einen Umstieg auf freie Software?

Was bei uns dafür gesprochen hat, ist erstens natürlich der Preis; zweitens ist ein wichtiger Punkt, finde ich auch, die Administrierbarkeit. Also jetzt haben wird das vom ZID drauf gespielt, Kubuntu, mit $HOME Verzeichnissen am NFS Server. Es ist zum Beispiel so, wenn ein Linux Computer eingeht, dann stellst du den weg, stellst einen neuen PC her, sagst ihm, welcher Client er werden soll, drehst ihn auf, der installiert sich, der User loggt sich ein und es hat sich für ihn nichts geändert. Probier' das mal konsistent in einem Windowsumfeld zu machen, das ist NICHT leicht. Du musst zwei Dinge machen, du musst die Leute konsequent dazu bringen, dass sie auf ein Netzlaufwerk alles speichern und du musst aufpassen; sie haben dann ihr eigenes Profil, das wird beim Einloggen zwischen Server und Client hin und her gespielt, mit dem hast unheimlich viel Probleme, wenn zu dem Zeitpunkt, zu dem der User ausloggt, der Server gerade nicht erreichbar ist, dann wird das Profil lokal gespeichert und dann sagt der dir, das lokale Profil ist jünger als das vom Server und fangt an, es nie wieder zu holen und mit dem lokalen zu arbeiten und lauter so Schmarren. Das ist die ganze Zeit so ... unter Linux ist das so problemlos und wir haben auch schon einen automatischen Installer gehabt. Also in Wahrheit: von der Administration her ist Linux viel viel angenehmer. Wir haben sogar das Dualboot im Pc-Labor so eingerichtet gehabt.

Wir haben zuerst Linux installiert und aus Linux heraus haben wir dann ein Windowsimage installiert. Das wäre mit Windows so nicht gegangen und du musst Windows auch ziemlich verbiegen, dass du das schaffst, aus einem Image das zu machen, wegen den Lizenz-keys, das macht natürlich viele Probleme. Das alles so einzurichten war schon einiges an Arbeit, aber es war eine einmalige Arbeit und es ist bei uns so gewesen: Es hat einige Leute gegeben, die ganz gut Shellscripting konnten und damit hat man eigentlich ein super Tool an der Hand, das alles kann, was du haben musst. Es war halt dann so, dass zu dem Zeitpunkt, als der ZID gesagt hat, er macht das, haben wir es aus der Hand gegeben. Der ZID bereitet jetzt Kubuntu als Distribution auf. Wir haben 10 Jahre lang Redhat laufen gehabt - das hat sich schon ausgezahlt.

8) Mit welchen Schwierigkeiten hat man bei einer Migration zu rechnen? (Verwaltung, Lehrende oder Lernende)

Das hängt davon ab, ob es gewisse Programme gibt oder nicht. Also z.B. bei Mathematica ist das Angenehme, das gibt es für Windows, Linux und Mac. Bei unseren Desktops an der Uni, das sind wechselweise 50 50 Windows-Linux, ist Mathematica auf allen installiert. Auf den Linux Installationen sogar automatisch, weil du da auch vernünftig automatisch installieren lassen kannst. Was natürlich die Probleme sind: du brauchst ein funktionierendes OpenOffice, weil aus mir nicht nachvollziehbaren Gründen die Windowsbenutzer darauf bestehen, ein Office zu haben, ich versteh das ehrlich gesagt nicht, weil ich brauche es z.B. kaum ... die Leute glauben, wenn sie eine formatierte Tabelle machen wollen, brauchen sie Excel, ich glaub aber, das das Irrtümer sind. Unsere Installation ist leider die "longtermsupport" ubuntu 6.06, dapper drake, die LTS Version. Die letzte LTS Version .. long term support ist nicht für Kubuntu 8.04 gemacht worden, weil KDE4.0 drinnen war. Jetzt sitzen wir auf dem Schmarren und das erwähnte OpenOffice funktioniert absolut unbefriedigend, gerade über NFS.

(Wieso nicht einfach die Version mit Gnome und LTS?)
Wir haben damals Leute auf Redhat umgestellt, mit dem KDE, weil wir Leute gehabt haben, die sich mit dem KDE gut ausgekannt haben, mit Gnome nicht. Dann hat man die von Windows auf KDE umgestellt und jetzt sollst du denen nach drei Jahren sagen: so jetzt stellen wir den Desktop auf Gnome um... das geht nicht gut.

Mir ist es völlig egal, ich habe jetzt das Ubuntu zuhause. Ich habe früher Redhat benutzt, habe den Standard Gnome Desktop; mir ist es ganz egal, was für einen Desktop ich habe, die können alle so viel mehr als ich brauche. Was natürlich schon ein Problem ist, wenn du anfängst Tastenkombinationen zu lernen und zwischen KDE und Gnome wechselst, dann passiert auf dem Desktop irgendwas.

9) Ist die Möglichkeit, die Funktionsweise eines Programms zu verändern, relevant für Bildungseinrichtungen? Informatikunterricht? Warum? Auf welche Weise?

Also ich denke mir, nicht in Termen des Betriebssystems selbst ... Die Software, die bei uns wirklich angegriffen wird, ist Spezialsoftware, die unabhängig davon ist, also zum Beispiel der Hermann Schichl nimmt ein Programm für globale Optimierung her und verändert das irgendwie, aber das ist unabhängig von dem, was mit Linux daherkommt oder mit Windows, das machen schon auch Studierende. Allerdings programmieren die dann typischerweise nicht an einem Firefox herum, sondern halt an irgendeinem Library für globale Optimierung. Wo wir diese Freiheit allerdings schon benutzt haben damals, war auf Grund der administrativen Aufgaben, also bei Linux war es z.B. leicht die Installation zu verändern, weil u.a. der ganze Bootprozess nur aus Shellscripts besteht, die man einfach so lesen kann, dann weiß man, was passiert und kann es für seine Bedürfnisse so ändern, dass es das tut was man haben will. Der gesamte Start, der „init“ Prozess, liest das erste Shellscript und die werden dann halt nach der Reihe ausgeführt. Das haben wir damals verändert und angepasst, aber den Bootprozess verändern kann man unter Windows wahrscheinlich auch, es ist halt weniger angenehm.

In den echten Code kann man aus einer Lehrsicht heraus kaum rein schauen. Ich hab mal spaßeshalber ein bisschen den Code von den Commandlinetools angeschaut, so was wie „cat“ oder so, das ist ja unlesbar. Ich glaube, aus Optimierungsgründen wird da in eine Richtung gegangen, die didaktisch nicht nachvollziehbar ist.

Wirklich da rein steigen tut man wohl selbst als Informatiker auf der TU nicht so weit, dass man auf diese Ebene in einer Lehrveranstaltung geht, wo du wirklich den Code selbst bearbeitest. Also da wirst du auch eher irgendeine Referenz-Implementierung nehmen, die halt ein bisschen langsamer ist als die tatsächliche, die dir aber die Prinzipien zeigt, weil sonst gehst du in technischen Details unter.
Weil es einfach viel zu schwierig ist, bis du ein Stück so eines Codes wirklich verstanden hättest, dass du ihn ändern könntest. Das kannst vielleicht im Rahmen einer Fachbereichsarbeit mit einem "Freak" machen. Den kannst du vielleicht dazu bringen, dass er das im eingeschränkten Rahmen macht, der kann sich aber genauso einen Laptop hernehmen und sich selber Linux drauf spielen und damit spielen.

10) Könnten Sie in wenigen Worten erläutern, warum offene Schnittstellen und Standards von Bedeutung sind? (in inwiefern ist das für eine Schule wichtig?)

(Anm. Frage wurde auf Grund vager Formulierung zu vorliegender Form verändert)

[...] Grundsätzlich.. absolut wichtig! [...] Zum Formatproblem: Ich glaube, das sind so Massenprozesse. Wenn alle .doc machen, machen alle anderen auch .doc und wenn nicht, dann halt nicht, aber das kann man schon auch ändern. Ich glaub allerdings nicht, dass das viel verändern würde.

11) Ist die illegale Vervielfältigung von Software ein Problem, mit dem sich Schulen auseinandersetzen müssen?

Man kann den Leuten sagen, dass es strafrechtlich relevant ist. Ich weiß es nicht, es ist sicher ein Punkt, der auftaucht; es ist schwer, ihn zu behandeln. Ich bin weder der Moralist, der sagt, man muss das auf jeden Fall zahlen und es wäre wahnsinniger Diebstahl noch kann ich jemandem dazu raten, das zu tun. Das sehe ich in gewisser Weise nicht ganz als meinen Zuständigkeitsbereich. Wenn du lehrst, musst du natürlich die Möglichkeit bieten, dass die Leute das in irgendeiner Form benutzen; ob es wirklich zu Hause sein muss, ist eine andere Frage. Aber in Schulen weiß ich nicht, wie man das macht.

Aber wenn man z.B. so etwas wie eine Ganztagsschule hätte, dann könnte man sich schon auf diesen Standpunkt stellen, wenn sie diese Hausübung machen müssen, dann sollen sie es am Nachmittag machen, wenn sie in der Schule sind. Man muss natürlich die Möglichkeit bieten, das kostet Geld und das musst du in die Hand nehmen, ob du jetzt in der Praxis wirklich so viel Geld aufwenden würdest und sagst du kaufst eine Gesamtlizenz, das weiß ich nicht. Weitergabe von Software unter der Hand an die Schüler ist natürlich problematisch. Das würde ich nicht machen. Wenn es wirklich an solche Bereiche geht wie Informatik und du machst dort wirklich was, im Sinne von programmieren oder sonst was, dann denke ich, kann man sich schon auf freie Lösungen beschränken. Also wenn es z.B. um das Programmieren geht, was spricht dagegen, Eclipse und Java zu nehmen, das jeder auf jeder Plattform einfach benutzen kann. Ich sehe keinen Sinn darin, wenn es nur um Entwicklung geht und nicht um Spezialisiertes, jetzt Tools zu verwenden, die nicht jeder haben kann. An der Fachhochschule, an der ich auch unterrichte, ist halt das Problem, dass wir proprietäre Hardware mit proprietärer Software zu deren Steuerung kriegen und damit absolut an diese gebunden sind. Du kannst maximal mit Emulation arbeiten oder mit virtuellen "Windosen" spielen, aber das Programm brauchst halt einfach und WINE ist auch kein Spass. Also ich versteh jeden, der dann sagt: Gut, dann muss ich halt mit Windows arbeiten.

12) Wie sollte im schulischen Kontext die Software-spezifische Ausbildung junger Menschen gestaltet sein?

Ich denke mir, wenn man in einer Schule Leuten Software generell beibringt, sollte man sich ganz grundsätzlich auf Grundfunktionen beziehen. Ich bin noch aus einer Generation, wo Leute viel mit dem Computer zu tun hatten, die das auch interessiert hat. Die Zeit ist längst vorbei. Die Leute können alle chatten usw., aber sie wissen nicht mehr, was sie tun. Als Bildungsauftrag würde ich es also am ehesten sehen, es passiert ja auch viel Schmarren damit, da werden die Leute furchtbar aufs Kreuz gelegt - über Softwaretricks. Ich denke, eines der wichtigsten Dinge wäre es, den Leuten Grundfunktionalitäten und Grundkonzepte zu zeigen; also wenn ich in einem Browser in ein Fenster hinein tippe, was das alles sein kann.

Was ist https im Unterschied zu http, solche Sachen, also wirklich grundlegende Dinge, um den Leuten zu zeigen, ob das plausibel ist, was da steht; kann das technisch überhaupt stimmen oder will mich da wer auf den Arm nehmen. Die Leute sind natürlich unsicher in der Frage, denn sie wissen ja nicht, was sie tun. Dies ein bisschen zu stärken wäre eine wichtige Aufgabe. Ich finde, konkrete Ausbildung auf eine bestimmte Software ist nicht Aufgabe der Schule, eher allgemein zu bilden. Ich würde Textverarbeitung gar nicht unterrichten. Wozu sollen die Schüler das lernen? Die sollen sich hinsetzen und es ausprobieren, wenn sie es wirklich brauchen. Aber die Frage ist auch, was sind mögliche Karrieren von Schülern, die man sich erhofft, wenn sie z.B. studieren gehen, was machen sie mit einer Textverarbeitung? Sie schreiben Texte hinein.. na und.. und dann machen sie einen Text fett. Wer studiert und das nicht kapiert, der ist ein wenig deplatziert. Man sollte den Leuten ganz grundsätzliche Dinge beibringen. Natürlich muss man sie bei der Stange halten und etwas machen, was ihnen vielleicht sexy erscheint, aber ich denke man sollte wirklich darauf achten, Grundlegendes zu unterrichten. Schon auf einem rudimentären Level, sodass sie mal verstehen, was in einem Netzwerk lokal und was remote ist. Kann es sein, dass das, was ich auf meinem Computer hier sehe, etwas ist, das über das Netz funktioniert? Das sind ganz wichtige Dinge; oder man könnte ihnen beibringen, was man damit anstellen kann, z.B. Google-Analytics: wenn man z.B. in einer Kaskade von Servern die Google-Analytics einsetzt, surft und zwischen drin einmal den Gmail-Account anschaut, dann hat Google zu deinem Namen deinen Surfverlauf gespeichert. So etwas zu wissen, das wäre wichtig und nicht wie man Schrift formatiert.

13) Kann es die Aufgabe einer Schule sein, wirtschaftlichen Monopolen entgegen zu wirken?

Nein, das wird sie auch nicht schaffen. Selbst wenn du die Leute jetzt top im Umgang mit etwas anderem ausbildest, was schon mal fraglich ist, ob du das schaffst; selbst wenn du das schaffst, hast du dann natürlich das Problem, dass sie dann ja nicht auf einmal zu zwanzigst in einen Betrieb kommen und sagen, wir wollen aber alle Linux, sondern sie kommen vereinzelt irgendwo hin und dann sagt wer, nein, bei uns ist alles auf Windows .. und basta.

14) Gibt es etwas zu diesem Thema, das wir noch nicht besprochen haben, aber noch gesagt werden sollte?

Was man vielleicht sagen sollte ist, dass man ein bisschen entspannter an das Thema herangehen kann, wenn über Umstellungen debattiert wird. Oft wird viel zu evangelistisch begonnen zu argumentieren. Wenn man die Idee hat, dass das geht, muss man den Leuten zeigen, dass die Desktops mittlerweile wirklich gut funktionieren, dass es in der Administration im Prinzip einfacher ist und dass es heutzutage wirklich jeder für das, was er so braucht, benutzen kann. Dann kann man das probieren, aber ob das wirklich funktioniert oder nicht, ist eine andere Sache. Aber es ist ja nicht so, dass Windows in Wahrheit praktisch ist, das ist es ja überhaupt nicht, wenn man z.B. probiert unter Windows etwas zu installieren, ist das ja nicht lustig. Unter Ubuntu z.B. hast du am Desktop ein Suchfeld, da kannst du einen Begriff eingeben, dann durchsucht er die Paketliste nach dem Begriff und du drückst auf "installieren" .. das war es. Ich finde, das ist in vielerlei Hinsicht einfacher und das muss man herzeigen, dann kann man schauen, was passiert. Bei den Mathematikern z.B. ist es sehr gespalten. Manche sind technisch sehr versiert, weil sie mit diesen Dingen auch zu tun haben, manche sind von der Computerbenutzung her in gewisser Weise Geisteswissenschaftler und manche von diesen arbeiten unter Linux und es ist überhaupt kein Problem, wenn sie sich daran gewöhnen. Außerdem ist remote Administration von einem Linuxsystem viel einfacher. Die Administration macht sehr viel aus. Ich glaube auch, wenn man es mit einer konsequenten Politik durchzieht, und Neuerungen wirklich in das System einbaut, sodass es auch automatisch wieder installiert würde - wenn man das konsequent macht, hat man irgendwann wirklich viel weniger Arbeit.

8.2.2 MMag. Rene Schwarzinger

Steckbrief:
Studium Informatik und Mathematik (Lehramt)
Studium Informatikmanagement
IT – Manager am BG RechteKremszeile

Wie kam es am BG RechteKremszeile zur Migration auf Linux und freie Software?

Es war so, wir haben einen wirklich sehr engagierten und sehr kompetenten Kollegen an der Schule, das ist Klaus Misof. Er ist sehr versiert in dem Bereich, obwohl er gar nicht Informatik unterrichtet; er ist einfach der Netzwerkbetreuer, der Ober-IT-Kustode, er ist Mathematiker und Physiker. Er kommt eigentlich aus der Forschung, ist überall in der Welt herum gekommen, hat sich aus privaten Gründen dann für die Schule entschieden und hat auch aus der Wissenschaft viel Kontakt mit Unix und anderen Betriebssystemen gehabt und ist halt auf dieser Schiene schon gekommen. Er ist also nicht der typische Windowsmensch, der da schon vorbelastet ist und nichts anderes kennen gelernt hat; er hat einfach schon Erfahrung gesammelt auf Grund seiner bisherigen Tätigkeit. Es war natürlich so, dass bis vor 2006, das Fach Informatik von den Lehrern eher schwach besetzt war an der Schule, das heißt, es hat keine geprüften gegeben. Ich bin da einer der ersten überhaupt, ich bin im Prinzip einer der ersten, die an einer technischen UNI die Prüfung zum Lehramt Informatik gemacht haben. Also von dem her hat sich das geschickt ergeben. Ich bin auf die Schule gekommen und er hat dann einmal so gefragt, was für Ideen ich habe und hat auch seine Ideen mal geäußert. Ich hab im Prinzip nicht mehr Linux Vorkenntnisse gehabt als irgendein anderer, aber ich war immer sehr offen; ich hab zum Schluss des Studiums dann zu meinem Windowsrechner noch einen Applerechner gekauft, ich hab mit Linux keine Berührungsängste gehabt. Er hat also gemeint, es wäre eine Überlegung bzw. Idee, für das Schulwesen das auf Linuxbasis zu machen.

Die Idee ist im Herbst 2006 entstanden; wir haben uns sehr schnell geeinigt, dass es eigentlich Sinn macht und wenn man die gewissen Rahmenbedingungen schafft, auf die ich noch näher eingehen werde, dass es auch umsetzbar ist, wobei uns von Anfang an bewusst war, dass es mit Schwierigkeiten verbunden sein kann, seien es Schwierigkeiten technischer Natur und vor allem, was als größeres Problem aufscheint, sind Schwierigkeiten in der Kollegenschaft; das war zumindest die Sorge von uns.

Da ist ein Punkt, der ganz wichtig ist: Im Vergleich zu anderen Schulen war es bei uns so, dass die Nutzung der Informatiksäle seitens der anderen Unterrichtsfächer eher noch gering war. Das heißt, wir hatten es natürlich leichter, gewisse Kollegen, gewisse Gewohnheiten umzuschulen; das ist ein Punkt, der das sicher begünstigt hat, es ist uns daher leichter gefallen. Wie gesagt, es war dann ca. 2006, als wir uns überlegt haben, was sind die ganzen Notwendigkeiten, die wir in der Schule haben und wo wollen wir eigentlich hin. Wir sind dann zu dem Entschluss gekommen, dass die vorhandene Linuxdistribution unseren Anforderungen nicht zu 100% entspricht. Unsere Distribution ist nicht die einzige in Österreich. „Desktop Open Server For Education“ aus Weiz in der Steiermark, die wird z.B. vom Bundesministerium gefördert, also es gibt so etwas schon. In Tirol hat es so etwas gegeben oder das deutsche Arktur, das sind alles Linuxdistributionen, die sich um die Schule drehen. Uns war wichtig, dass diese Linuxdistribution nicht nur an der Schule eingesetzt werden kann, sondern auch von den Schülern zuhause. Das hat es einfach nicht gegeben, das heißt, Stichwort Livesystem. Bei uns hat das gewechselt eine gewisse Zeit, wir haben zuerst versucht ein Grundsystem zu finden, das wir adaptieren konnten, also anpassen, das wirklich auf älteren Rechnern läuft. Wir haben auch Erhebungen gemacht, wie schaut es bei Schülern zuhause aus, und die haben halt häufig den Rechner zuhause im Einsatz, den die Mama oder der Papa hergegeben hat, weil er nicht mehr so gut funktioniert, weil er schon sieben Jahre alt ist und nur 256 MB Speicher hat. So stellten wir die Bedingung, dass es eine sehr schlanke Distribution sein sollte, sonst haben wir ein Problem. Wir haben also gesagt, der Desktop sollte Xfce sein, nicht Gnome und auch nicht KDE, weil der ressourcenschonender ist und auch vom System her soll es ressourcenschonend aufgebaut sein. Das war einmal ein Punkt: Die Auswahl einer vorhandenen Distribution, die am annäherndsten unseren Bedürfnissen entspricht. Edubuntu war im Gespräch, es war Dreamlinux im Gespräch, für das haben wir uns auch entschieden, es zu nehmen, ca. zwei, drei waren in der engeren Auswahl.

Als wir überlegt haben, diese zu adaptieren, haben wir dann zufällig in der Zeitschrift „Linuxuser“ gesehen, dass es eine brasilianische Distribution gibt; das war Dreamlinux, die einfach extrem ressourcenschonend ist. Uns war wichtig, dass es auf jeden Fall Debian basierend sein sollte, weil einfach viele Pakete dafür existieren; andere wie SuSE schieden aus, weil es sich herausgestellt hat, dass es damit viel mehr Troubles gibt. Dreamlinux basiert also auf Debian, aber auch auf anderen. Klaus Misof ist dann zu dem Standpunkt gekommen, dass dies die Adaptierung gewisser Sachen schwieriger macht, weil die selbst schon so eine Mischung gehabt haben. Der nächste Schritt war dann, dass wir auf Ubuntu gewechselt haben. Wir haben dann natürlich Dreamlinuxteile weiterhin in unsere Ubuntu-Variante integriert und andere Sachen, so weit es möglich war. Mittlerweile sind wir einfach auf ein reines Debiansystem zurückgegangen und haben das genau an unsere Vorstellungen adaptiert. Debian 4 ist momentan die aktuelle Version. Es gibt auch ein Debian „live“ Projekt, wo wir auch schon das Livesystem drinnen haben. Dieses ist auch ca. gleich aufgebaut wie das installierte System. Das ist ein Vorteil zu anderen Livesystemen.

Wir sind ein kleines Entwicklerteam; wir sind jetzt einmal drei Leute und es wird überlegt, einen Vierten dazu zu nehmen. Wir schauen, dass wir jedes halbe Jahr eine Release herausbringen, öfter ist nicht notwendig. Es würde im Schulsystem sogar einjährlich reichen und natürlich schleichen sich immer wieder Fehler rein, oder wir kommen sagen; das wollen wir jetzt auf jeden Fall noch haben, deshalb haben wir versucht, es halbjährlich zu machen. Sinnvoll wäre es natürlich in den Sommerferien, wenn Administratoren sich überlegen, ein neues System zu installieren und nicht zum Beispiel im Oktober. Aber die Entwicklung ist sicher zeitaufwändig, wobei nur dann intensiv, wenn wir eine neue Distribution heraus bringen. Diesen Aufwand müssen natürlich andere nicht betreiben, die diese einfach nur einsetzen möchten. Das ist von unserer Seite ein gewisser Enthusiasmus, gewisse Freizeit, die da geopfert wird. Es kann nie der Sinn sein, dass jede Schule ihr eigenes Linux baut, das wäre kontraproduktiv. Wir glauben nicht, dass wir die einzige sinnvolle Lösung haben, sondern wir versuchen einfach, eine Lösung bereit zu stellen, mit der die meisten Schulen im deutschsprachigen Raum sinnvoll arbeiten können, damit und es nicht nur auf die Schule beschränkt ist, sondern auch die Schüler zuhause was davon haben.

Das ist unser wichtiger Punkt, das unterscheidet uns z.B. von dem Weizer Projekt. Deren Distribution müssen die Schüler zuhause installieren, und dass dies nicht so einfach ist, da kommt man recht bald drauf. Deswegen ist bei uns der Schwerpunkt der Entwicklung auf dem live USB-Stick, wo die Schüler ein voll funktionsfähiges Betriebssystem auf ihrem USB-Stick haben und nichts mehr installieren müssen, nur mehr den USB-Stick anstecken. Zur Zeit ist es so, jeder kann seinen USB-Stick mit dieser aktuellen Version von LinuxAdvanced bestücken. In der Praxis schaut das so aus: Ich bespiele es einfach den Schülern und gib es ihnen am nächsten Tag zurück. Sie müssen sich vorher die Daten sichern, zehn Minuten hab ich Aufwand damit. Das ist jetzt so, in Zukunft sollen sie das natürlich selber machen können, aber der Vorteil ist natürlich, sie haben zuhause am Stick Betriebssystem und Daten auf einem Medium. Sie brauchen am Rechner nichts mehr zu installieren und es funktioniert genauso schnell, wie wenn es installiert wäre.

Ich hab es bis jetzt nur jenen Schülern drauf gestellt, die vorher schon gesagt haben, sie haben einen Computer, der nicht älter ist als zwei Jahre; da ist das normal voreingestellt auch schon. Grundsätzlich haben wir das immer geschafft und sonst haben wir halt ein bisschen erklärt, manchmal ein wenig umständlicher, manchmal nicht, aber es geht. In Zukunft ist es so, dass wir vorhaben, vom installierten System im Informatiksaal wegzugehen, und ab nächstem Jahr komplett vom USB-Stick auch im Informatiksaal zu fahren. Dann ist natürlich so, dass wir das schon mit den Schülern gemeinsam machen. Auch USB-Ladestationen sind geplant, wo zum Beispiel die Schüler, wenn ihr System einmal nicht funktioniert, einfach dieses wieder zurück spielen und in fünf bis sieben Minuten wieder ein komplettes System in der Hand haben. Mit diesem können sie auch zuhause arbeiten, müssen aber nicht, das ist wichtig; aber in der Schule müssen sie natürlich damit arbeiten und den USB-Stick brauchen sie natürlich sowieso. Wir haben geschaut, dass jeder Schüler einen USB-Stick hat, mindestens 2 GB, die sind auch recht billig, zwei GB bekommt man heute um 10 Euro oder darunter. Das Booten vom USB-Stick dauert vielleicht ein paar Sekunden länger und im Betrieb merkt man überhaupt keinen Unterschied. Natürlich kann die Schreib- und Lesegeschwindigkeit von USB-Sticks unterschiedlich sein, aber es ist praktisch kaum bemerkbar, dass es langsamer wäre im Vergleich zu einer live CD oder live DVD. Damit haben wir angefangen und im ersten Jahr auch an alle Schüler verteilt. Das haben manche Schüler genutzt, aber das war natürlich mehr zum Kennenlernen gedacht; denn für den effektiven Betrieb kommt das nicht in Frage. Am USB-Stick ist dieses Problem nicht gegeben.

Auch da haben wir vor, zwei Arten anzubieten: Die eine ist natürlich, der Schüler kann im Prinzip nichts verändern am System, d.h. beim nächsten Mal Starten schaut es wieder gleich aus. Das ist eher für Unterstufenschüler gedacht. Für Oberstufenschüler gibt es dann eine Version, bei der man alles beeinflussen kann. Das heißt, alles, was man zusätzlich installiert, ist beim nächsten Hochfahren auch wieder oben. In diese zwei Richtungen werden wir es vielleicht aufteilen. Der Vorteil ist, wenn man es mit einem frisch aufgesetzten Windows vergleicht, es ist an Programmen schon alles oben, was man braucht. Spiele haben wir bewusst weggelassen, die Oberstufenschüler können sich dann die Spiele drauf installieren. Aber es gibt kein Programm, das die Schüler brauchen würden, das nicht schon oben ist. Von dem her ist diese Notwendigkeit in der Unterstufe gar nicht gegeben, dass sie irgendwas dazu installieren müssten. Wir haben in der Schule 2007 zum ersten Mal Linux voll im Einsatz gehabt: in allen Informatiksälen, im Konferenzzimmer, in den Spezialsälen; nur in der Verwaltung anfangs nicht, das haben wir aber jetzt auch schon langsam umgestellt. Das ist aber vielleicht auch ein eigenes Kapitel. Das heißt, wir haben uns 2006 geeinigt, wir haben eine Vorlaufzeit von einem guten halben Jahr für uns selber gehabt und Klaus Misof hat die erste Version sehr schnell angepasst. Diese war natürlich noch nicht so ausgereift, aber es war schon so, dass sie für den Betrieb im Informatiksaal geeignet war. Das war aber nur das Technische; das Wichtigste war aber, dass das ganze Akzeptanz findet. Die Kollegen muss man rechtzeitig darüber informieren und vor allem sie auch einschulen. Das heißt, vor allem auf die Programme, die anders sind. Wir haben versucht, den Schwerpunkt auf OpenOffice zu setzen, das haben wir vorher schon unter Windows dazu installiert. Das heißt, wir haben es natürlich parallel auch schon gehabt, es war ein schrittweiser Umstieg, das ist wichtig. Da haben wir ihnen wirklich laufend schulinterne Lehrerfortbildungskurse angeboten, das war noch, bevor sie Linux überhaupt gesehen haben. Da haben wir gesagt, wir werden das umstellen, haben ihnen erklärt, warum das Sinn macht; dass es auch Kurse gibt und dass es nachher kein Problem sein wird und dass natürlich auch im Bedarfsfalle immer individuelle Betreuung stattfinden würde. Diejenigen, die beabsichtigt haben, die Computer in ihrem Fach zu nutzen, haben die Kurse besucht. Wir haben vor dem Sommer 2007 den ersten Linux-Kurs angeboten, um zu zeigen, wie das System aussehen würde, das die Lehrer im Herbst dann verwenden könnten. Wir haben ihnen gezeigt, wie sie sich anmelden müssen; wie schaut das mit dem Benutzernamen aus; wie funktioniert der Dateimanager; wie können sie wieder ihr Office-Programm starten; wie können sie speichern.

Das haben wir ihnen alles mal gezeigt, haben ihnen quasi die Angst genommen vor der Umstellung, rechtzeitig bevor sie noch wirklich hinein gestoßen worden sind, mit was anderem zu arbeiten. Wir haben die Schulungsangebote wirklich intensiv betrieben, das waren viele Kurse. Wir haben den Faden dann auch noch weiter gesponnen, nicht nur für Office, sondern auch für Bildbearbeitung; ein bisschen Gimp gezeigt denjenigen, die es interessiert hat und ihnen gezeigt, dass es gar nicht viel anders ist als die Sachen, die sie vorher schon gekannt haben wie z.B. Photoshop oder irgendwas. Wir haben einfach versucht, das wirklich breit abzudecken; vor allem das Office, das haben wir mehrmals gemacht. Also ich habe, glaube ich, 3-mal einen OpenOffice-Writer-Kurs und zwei Impress-Kurse gehalten. Die meisten Kurse davon sind am späteren Nachmittag angesetzt worden, wo dann natürlich auch die Kollegen ihre Freizeit geopfert haben, aber es ist einigermaßen gut angenommen worden. Es war zwar nicht immer gleich die Masse bei jedem Kurs, aber durch die Fülle der Kurse haben wir das Wissen, das für die Umstellung notwendig war, insgesamt eigentlich auf viele Kollegen übertragen können. Und wir haben ihnen auch unsere Gründe für das Ganze erklärt, warum vielleicht das in der Schule einfach Sinn macht, dass es vielleicht sicherer ist, Viren-Problematiken etc. Wir haben wirklich sehr viel gemacht dafür, das stimmt schon, aber sie haben es nicht nur akzeptiert, sie haben es wirklich unterstützt, geklatscht in der Konferenz und sie waren begeistert dafür. Das hat mich selbst überrascht, mit dieser positiven Stimmung hätte ich selber nicht gerechnet, weil es in der Kollegenschaft oft das größere Problem ist als vielleicht manche technische Umstellung.

Wir sind aktiv drei Informatiklehrer, ein paar sind in Karenzzeit momentan. Es kann nicht jeder ein Unterstützer sein, aber von Informatikern erwartet man es zumindest schon, bei anderen Kollegen kann man das diskutieren. Für mich war es klar, dass Windows nicht das alleinige Allheilmittel ist, sondern, dass man von Fall zu Fall unterscheiden muss, ist es hier sinnvoll oder nicht und dementsprechend dann urteilen. Da habe ich kein Problem gehabt, auch wenn es für mich eigentlich viel mehr Aufwand war.

Ich habe mich selbst mit Linux mehr beschäftigen müssen, habe viel Freizeit dafür geopfert, aber insgesamt ist das einfach eine Sinnfrage für mich gewesen. Was noch zu erwähnen wäre, ist, dass wir sind eine der wenigen Schulen sind, die zwei geprüfte Informatiklehrer hat. Das ist sicher auch etwas, was begünstigend ist, obwohl viele Schulen sicherlich viele engagierte Informatiklehrer haben und das ebenfalls kein Problem wäre. Wir haben einen Schwerpunkt in Informatik: In der Unterstufe von der ersten bis zur vierten Klasse eine Stunde, in der Oberstufe dann zwei Stunden bis zur achten Klasse.

In der Unterstufe ist es so, dass es kaum auffällt, ob man mit Linux oder mit Windows arbeitet, weil der Hauptschwerpunkt dort das Office-Management ist. Da haben wir jetzt das OpenOffice, das können sie auch unter Windows zuhause installieren, also haben sie dasselbe Programm auf einem anderen Betriebssystem. Ein anderer Schwerpunkt neben Office wäre das Internet und Dateimanagement und solche Geschichten, kurz; Office-Management, da ist wichtig, dass man Software einsetzt, die auf allen Systemen lauffähig ist. Wir haben in der Unterstufe OpenOffice im Einsatz, Firefox, Thunderbird, Dateimanager sind eh überall ähnlich, Audiobearbeitung Audacity, Bildbearbeitung Gimp. VLC als Mediaplayer. Das ist mir persönlich auch viel viel wichtiger als jetzt den Schwerpunkt aufs Betriebssystem zu setzen. Dass Schüler eine Software zur Verfügung gestellt kriegen, die sie zuhause problemlos installieren können; die sie legal installieren können; die sie auch auf einem anderen Betriebssystem installieren können, wenn sie mal wechseln müssen. Spätestens im Studium werden sie mit Linux konfrontiert werden. Auch die Firmen werden sicher die Vorteile abwägen, vor allem, wenn vielleicht neue Generationen mit mehr alternativen Fähigkeiten herauskommen, die weniger Vorurteile haben. Von dem her ist mir wichtig, dass die Software plattformunabhängig ist. Spezielle Lernsoftware?

Es gibt zum Beispiel in Englisch eine Lernsoftware, die heißt „you and me“. Die ist so selten eingesetzt worden, dass es am Anfang den Lehrern gar nicht aufgefallen wäre, wenn es nicht gegangen wäre. Aber es gibt unter Linux WINE, um Windowsprogramme laufen zu lassen, z.B. diese Lernsoftware kann problemlos unter Linux verwendet werden. Dann haben wir noch Biologiesoftware, das ist irgendeine spezial CD, die funktioniert mit WINE tadellos. Wir haben in unserer Distribution in LinuxAdvanced, das ist der Name, gewisse KDE Lernprogramme inkludiert, die in Anlehnung an Edubuntu übernommen worden sind.

Als Tipptrainer z.B. haben wir Ktouch; den haben wir dann natürlich auch mit eigenen Kursmodulen bestückt, damit es mit dem Buch, das wir bestellt haben, zusammen passt. Das sind immer so Erweiterungen. Dann sind noch einige kleine Tools dabei, die momentan in der Praxis eher weniger eingesetzt werden wie Kpercentage, Kbruch, usw. Man könnte schon einiges davon benutzen; es sind durchaus Tools dabei, die man didaktisch sinnvoll verwenden könnte, wobei ich jetzt aber weiß, dass diese noch kaum verwendet werden. Das hängt natürlich stark damit zusammen, inwieweit die Informatik als tägliches Unterrichtsmittel in den Gegenständen akzeptiert und angesehen wird.

Das ist einfach ein Prozess, der im gesamten Schulsystem erst kontinuierlich voranschreitet und es wird in vielen Schulen so sein, dass es nur ab und zu eingesetzt wird. Es wird dann auch immer mehr eingesetzt werden und dann werden vielleicht auch andere Anwendungsgebiete eine größere Rolle spielen.
Support? Grundsätzlich hat man in der Schule die Möglichkeit, das System selbst zu warten oder man gibt einer externen Firma den Auftrag. Bei uns ist das so, dass Klaus Misof das komplett übernommen hat, als er an die Schule gekommen ist. Er wird das in zwei Jahren oder so vielleicht übergeben und vielleicht mit meinem Kollegen aufteilen. Aber wir werden das wahrscheinlich im Haus belassen und sind natürlich sehr stark bemüht, das ganze System möglichst wartungsarm zu machen. Das heißt, unser Ziel ist es, dass vielleicht nächstes Jahr schon die Schüler mit ihrem eigenen USB-Stick in der Schule starten. Wir haben jetzt schon die neuen Rechner ohne CDROM Laufwerk gekauft, wir würden in Zukunft keine Festplatten mehr brauchen, wir könnten bei den Rechnerkosten sparen, wir könnten bei den Wartungskosten direkt sparen. Im Grunde soll das so ein p2p Netzwerk werden, wo jeder mit seinem USB-Stick hochfährt und jeder dann Freigaben auf seinem System freischaltet. Obwohl die Betriebssysteme nur vom USB-Stick fahren, wird in der nächsten Version integriert sein, dass solche Freigaben automatisch für die Schüler wieder aktiviert werden können. Wir wollen auch den Server so weit wie möglich abspecken. Wir haben zwar nachher noch einen Server, die Serverversion, die müssen wir noch fertig entwickeln. Da sind wir leider ein bisschen in Verzug gekommen, weil das Team natürlich eher klein ist. Der Server soll nachher hauptsächlich die Webseite beherbergen. Mailserver möchte ich auslagern lassen; ich sehe das nicht als Aufgabe jeder einzelnen Schule, einen Mailserver zu betreiben, da muss man natürlich eine sinnvolle Alternative suchen. Der Server soll dann auch noch ein Skript für Internetsperre laufen haben, sodass man saalspezifisch diese Sachen wegschalten kann. Es sollen natürlich Samba-Freigaben ermöglicht werden, sofern das notwendig ist, aber im Grunde genommen soll der Server möglichst wenig machen. Jetzt macht er noch wesentlich mehr. Momentan läuft da auch eine LDAP Datenbank, das werden wir ändern auf ein MYSQL System, weil es hier leichter ist ,gewisse Sachen dazu zu programmieren. Es ist LDAP auch einfach schwerer zu konfigurieren. Die $HOME Directories der Schüler liegen natürlich am Fileserver, der nimmt die Authentifizierung vor am LDAP, die kommen mittlerweile per NFS und nachher ist natürlich so, dass die /home Verzeichnisse am USB-Stick liegen.

Jetzt ist es so, man sagt ihnen zwar, sie sollen am USB-Stick speichern, damit sie die Daten zuhause haben. Aber manche vergessen den USB-Stick, dann speichern sie wieder am $HOME Directory, welches am Server liegt. Es ist ein bisschen inkonsequent, aber wenn sie natürlich auf den USB-Stick angewiesen sind, ist das anders. Wenn man den USB-Stick nicht hat, muss es natürlich Maßnahmen geben, aber das wird sich sehr schnell aufhören, wenn die Schüler sehen, sie können ohne nicht arbeiten. Die Mitnahme-Quote der USB-Sticks ist jetzt auch schon einigermaßen erfolgreich, so oft werden sie nicht vergessen. Zur Zeit sehen wir den USB-Stick als zentrales Medium, im Prinzip sind die Schüler aber nicht mit HighEnd-Geräten ausgestattet und wir haben die Aufgabe, auf die schwächsten Rücksicht zu nehmen. Die Schüler haben alte Rechner, wir legen ihnen nahe, das OpenOffice auch fürs Windows zu verwenden und spielen es in der Stunde gemeinsam im Unterricht auf den USB-Stick. Wir zeigen z.B. über ein Windowssystem, das wir virtuell laufen lassen, oder am Laptop, wie man es installiert; das ist keine große Herausforderung mehr. Wir schauen, dass die Schüler die Software für Windows beziehen können, mit der wir arbeiten. Jenen, die dann sagen, sie haben schon einen neuen Computer oder die sich ärgern, weil der Computer zuhause gerade nicht funktioniert, sagen wir, willst du nicht das probieren, mach es gleich so zu Hause und dann geht es vielleicht eine Stufe geschickter.
Wir sind hier ca. 50 Lehrer und ca. 500 Schüler; die Rechnerzahl ist im letzten Jahr stark gestiegen. Wir haben ca. 130 Rechner, ich hab jetzt aus der Privatwirtschaft 50 Rechner erhalten, die ca. 3,5 Jahre alt sind und wir haben keine hergegeben. Das heißt, wir haben mehr Möglichkeiten geschaffen.

Vista z.B. auf den Rechnern, die wir haben, wäre undenkbar. Man könnte natürlich Windows XP weiterlaufen lassen, das würde schon gehen, aber dass man immer aufs aktuellste System zurückgreifen kann, das wäre bei uns bei Windows natürlich nicht möglich. Das ist auch ein Grund. So kann man mit dem Fortschritt mehr mitgehen, weil ein Linuxsystem einfach ressourcenschonender umgeht als ein Windowssystem. Das ist sicher ein positiver Nebeneffekt. Ein Problem ist, dass manche zu sehr Angst haben und dass der Rückhalt von der Kollegenschaft nicht gegeben ist. Es ist eigentlich weniger das technische Problem als dieses. Da ist vielleicht einfach nur ein Informationsdefizit gegeben, woran man sicher ebenfalls arbeiten muss; weniger von unserer Seite aus, als viel mehr vom Landesschulrat z.B. dass man die Information, wie man so etwas betreibt und durchführen kann, weiter gibt und die Angst davor auch nehmen kann.

Der Kostenfaktor war bei uns immer im Hintergrund. Wir haben im Prinzip keine Lizenzkosten zu tragen, aber - wer sind wir? Wir sind wir nur Lehrer, die Angestellten an der Schule oder sind wir vielleicht sogar die Eltern, die Schüler, wir definieren unser „wir“ größer. Wir sind eine Schulgemeinschaft; unser Hauptklientel, die wichtigsten, sind die Schüler, nicht die Lehrer und von dem her ist es sehr wohl auch ein Kostenfaktor, der zum Tragen kommt. Wenn ich mal schätzen würde, wie viele Windowsversionen legal installiert sind, traue ich mich zu sagen, vielleicht 30 Prozent; aber von recht viel mehr geh ich nicht aus Dass die Schüler sehen, dass man einigermaßen leicht das Ganze auch legal betreiben kann; dass es sehr wichtig ist, auch hier einen legalen Weg zu gehen und vor allem, dass es wirklich sinnvolle Alternativen gibt, ist sicher auch ein Lerneffekt, der durch die Schule umgesetzt werden muss. Dass es nicht als selbstverständlich gilt, es einfach zu brennen, sondern dass es vielleicht besser ist, diese Alternative zu wählen, die auf meinen konkreten Anwendungsfall bezogen einfach für mich kostengünstiger und sinnvoller ist. Das ist natürlich ein Punkt, der zur Zeit für die Eltern sicherlich viel mehr eine Rolle spielt als für uns, was aber auch für das ganze Bildungssystem in Österreich sehr wohl zum Tragen hätte kommen können, wenn der Bund nicht z.B. diesen Sommer die Lizenzen für Betriebssystem und Office verlängert hätte. Das Betriebssystem zu verlängern war naheliegend, im Falle der Office-Lizenzen ist es für mich jedoch unverständlich.

Da gehen Millionen raus, die in anderen Bereichen fehlen und die der Bund einfach herschenkt. Sie haben sogar eine Office 2007 Lizenz angekauft, was ich persönlich auch nicht unbedingt als sinnvoll erachte. Das MS Office 2007 enthält ja kaum neue Funktionen, aber eine komplett geänderte Usability. Für Anwender, die schon Office kennen, ist der Schritt zu OpenOffice viel einfacher als der zu MS Office 2007, wobei das Konzept für Neueinsteiger grundsätzlich ja nicht zu kritisieren ist. Aber so schlecht ist das alte System auch wieder nicht, dass ich so großartige Vorteile sehe, die es rechtfertigen würden, jetzt Office 2007 in der Schule einzusetzen. Es sind zwar gewisse Sachen unter MS Office besser, aber es läuft unter Linux nicht und es läuft am Mac nur bedingt. Das ist eine eigens entwickelte Office-Version, nicht die idente, da gibt es auch dort Probleme.

Die HAKs sind auch sehr auf Microsoft Skills Training ausgelegt, kommt mir vor. Die einen begründen das damit, dass sie Produktschulungen für Microsoft betreiben müssen, weil es ja in der Wirtschaft eingesetzt wird. Es gibt aber Gegenargumente und die sind auch von anerkannten universitären Persönlichkeiten, die sagen, die Wirtschaft würde sehr wohl den Bedarf haben, mehr Linux, mehr alternative Office Programme einzusetzen. Wenn die Schüler besser ausgebildet wären, weniger Vorurteile hätten und vielleicht ein bisschen damit gearbeitet hätten, vorher schon in der Schule. Weil es ihnen oft nur aus diesen Gründen zu mühsam ist und sie daher bereit sind die Lizenzkosten zu zahlen. Es ist immer so eine zweigleisige Geschichte, die einen argumentieren so und die Anderen genau dagegen. Es wird sich da wohl in der Mitte irgendwo treffen. Zudem, die Ausbildung darf ja nie so sein, ich darf nicht Microsoft Word unterrichten oder OpenOffice Writer, ich muss Textverarbeitung unterrichten. Ich muss im Prinzip die Schüler soweit bringen, dass sie, wenn sie das eine können, das andere innerhalb einiger Tage genauso gut beherrschen; das muss das Ziel sein. Wir im Informatikunterricht sagen dann z.B. "da ist jetzt ein gewisser Unterschied zu Microsoft Word" wenn wir im Writer arbeiten usw. Dort, wo ich es weiß, sag ich es natürlich dazu. Es geht überhaupt nicht um Produktschulungen, sondern um übergeordnete Fähigkeiten und deswegen versteh ich nicht, warum ich jetzt eine spezielle Software im Unterricht brauche. Es geht darum, nicht auf hinklicken zu trainieren, sondern vielmehr auf Verständnis, vielleicht auch im Menü überhaupt etwas zu finden.

Dann ist in einem anderen Programm auch viel leichter etwas zu finden als wenn ich immer nur sage, klick dahin, klick dorthin. Es wäre sicherlich auch interessant, mal mit Abiword oder etwas anderem zu arbeiten, damit man sieht, das Programm heißt anders, aber es funktioniert eigentlich gleich.

1) Setzen Sie freie Software privat ein?

Privat habe ich zu Hause einen Mac stehen und einen Linuxrechner und das wird gemischt.. also ja! Mac ist jetzt natürlich nicht frei, aber von der Software, die ich installiert habe, setze ich hundert Prozent freie Software ein. Es gibt für mich privat kein Betätigungsfeld, wo ich keine freie Software finde, die es abdecken könnte.

2) Benötigen Sie Anwendungssoftware, die nicht oder nur unzureichend durch das Angebot freier Software gedeckt würde?

Nein.. das gibt es bei uns nicht. Es gibt Programme, gegen die sich manche mit Händen und Füßen wehren; wo manche sagen, das ist nicht so gut, weil sie sehr eingesessen sind in ein Programm, also z.B. DG Lehrer sind sehr schwer zu überzeugen, dass es z.B. Qcad gibt, das ebenfalls diese Funktion übernehmen könnte. Manche haben, glaub ich, den Eindruck, dass ein Programm, das hunderttausend Funktionalitäten besitzt, dass diese Vielzahl an Funktionalitäten der alleinige Grund sind es zu verwenden und sie sehen eigentlich nicht, dass sie neunundachtzigtausend Funktionen nie benutzt haben und das alternative Programm eigentlich die Funktionen, die sie wirklich nutzen, auch hätte. Bei uns arbeiten die Direktorin und die Sekretärin mittlerweile schon auf einem Dualbootsystem und ein Großteil der Verwaltungsprogramme, die wirklich für Windows oder Microsoft maßgeschneidert sind, sind auch unter Linux schon lauffähig, zum Teil mit WINE. Das Problem ist, dass manche Anwendungsprogramme, die unter Windows laufen; auch dort nicht ganz reibungslos sind. Die Stabilität ist grundsätzlich zu kritisieren, das hat wenig mit WINE zu tun. Es passieren leider auch Fehler in der Ausschreibung, wie das wieder bei einer Ausschreibung vor einem Jahr etwa war, wo wieder nicht darauf geachtet wurde, dass eine plattformunabhängige Software ausgeschrieben wird.

Es passieren einfach Fehler, die nicht sein müssten. Da kommen wir noch einmal auf den Kostenfaktor. Das Wienux ist auch daran gescheitert, weil sie geglaubt haben, alles ist gratis und freie Software ist eben nicht immer gratis. Die Münchner Stadtverwaltung hat für das gleiche Projekt, bei dem sie all das umgesetzt haben, was die Wiener als Probleme beklagt haben; die haben gesagt, das SAP funktioniert nicht.

Die Münchner haben es sehr wohl geschafft, dass es funktioniert, und sie haben auch andere Sachen geschafft; die haben dafür 14 Mitarbeiter in der Stadtverwaltung angestellt; das waren Informatiker oder Leute mit informatiknahen Ausbildungen. Zusätzlich haben sie kleine lokale Open Source Firmen angestellt und das Projekt funktioniert. Wenn man nichts hineinsteckt, kommt halt auch nur bescheiden was raus, das war in Wien das Problem. Die Dienstleistungen bleiben so auch im Land und gehen nicht raus. In Wien haben sie jetzt eine „Rückimmigration“ initiiert, was ja noch dümmer ist, da sie vorher schon fast eine erfolgreiche Migration geschafft haben und das ist absolut unverständlich. Das Problem ist hier auch, dass sehr stark einfach ein Microsoft Lobbyismus passiert, der von hoher Instanz auch offen zugegeben wird. Aber im Prinzip.. das Geld kann man gut brauchen, dann ist das andere wieder egal... dass das zwar nicht so zukunftsträchtig und wahrscheinlich trotz des ganzen Lobbyings noch viel teurer ist, wird halt unter den Tisch gekehrt.

3) Was halten Sie von diesem Statement?

Dieses Wissen ist natürlich nur für manche Personenkreise zugänglich, die auch das Know-how haben, aber wichtig ist, dass jene Personen, die das Know-how haben, auf dieses Wissen zurückgreifen können. Es profitieren ja auch andere indirekt davon. Es ist nicht so, dass nur derjenige den alleinigen Nutzen davon hat, der eben das Know-how hat wie bei uns z.B. der Klaus Misof, der wahrscheinlich überdurchschnittlich begabt ist in dieser Richtung. Vor allem anderen ist das aber auch die Reduzierung der Abhängigkeit von einzelnen Institutionen, die einfach eine Machtkarte ausspielen können. Von dem her ist natürlich auch wichtig, wenn Quellcode von Standard- von Basissoftware offen ist. Ich bin jetzt nicht der Verfechter, der sagt, alle Software der Welt muss unbedingt frei sein. Aber zumindest von jener Software, von der wir abhängig sind, ist es sicher sehr gut, wenn der Quellcode frei ist, weil einfach die Abhängigkeit dann nicht in dem Ausmaß gegeben ist. Gewisse technologische Entwicklungen können schneller vorangehen, wenn Sachen frei zugänglich sind, weil durch eine größere Community mehr passieren kann: Es entsteht mehr Konkurrenz, mehr freier Markt, mehr Wettbewerb. Das ist sicher ein Punkt, der auch Hersteller von proprietärer Software zu Innovationen drängt.

4) Sollten und können Überlegungen dieser Art bei der Wahl der Software für eine Schule von Bedeutung sein?

Die vier Freiheiten sind auch bei uns positiv zu beurteilen. Wir haben z.B. im ersten Jahr eine Live CD entwickelt und sie fünfhundert Mal vervielfältigt, oder der Schüler kann es zuhause vervielfältigen und teilt es dann seinem ganzen Dorf aus. Das ist nur mit freier Software möglich und bei proprietärer Software nicht erlaubt. Wenn wir jetzt von der ideologischen Seite aus gehen, dann sind wir auch gleich bei der Unterscheidung von dem Begriff von freier Software und Open Source Software, wo diejenigen die das ideologisch betrachten, die Freiheit der Gesellschaft und die Opensource Leute eher den Mehrwert, kollaborativer Entwicklung von Software sehen. Die einen sehen das nicht nur auf die Technik beschränkt, sondern auf das Soziale, Ethische umgemünzt. Ich persönlich bin einer, der grundsätzlich sagt, wo liegt eigentlich der Nutzen von einem Ding, das ich verwende. Ist es für mich die beste Lösung? Ich versuche an so etwas immer pragmatisch heranzugehen, aber grundsätzlich, wenn ich die Auswahl habe zwischen zwei Produkten, die für mich den gleichen Nutzen haben, kommt ganz klar eine freie Software zum Vorzug, auch aus ideologischen Gründen. Die Folgewirkungen sind vielleicht nicht so leicht abschätzbar. Die Schule ist im Prinzip auch eine Firma, auch wenn sie nicht als solche geführt wird, aber eine Schule muss genauso eine Kosten/Nutzen Rechnung anstellen und die wird zur Zeit einfach noch schlecht geführt. Aufgrund dieser müssen Entscheidungen, die getroffen wurden, revidiert werden. Wenn man eine allgemeinbildende Schule ist und eine gewisse Persönlichkeitsbildung, eine gewisse gesellschaftliche Entwicklung auch an die nächste Generation weitergeben möchte, ist das zumindest ein Thema, das nicht einfach so unter den Tisch gekehrt werden darf. Man muss sich überlegen: Ist es sinnvoll, wenn mir jetzt irgendwer vorgibt wie ich die Software einsetzen darf? Ich zahl schon dafür und er sagt, ich darf es nur auf einem Rechner einsetzen und es nur zu diesem Zweck einsetzen. Ist moralisch vertretbar, wenn solche Einschränkungen möglich sind, die durchaus nicht weither geholt sind. Microsoft hat z.B. in China jetzt für 10 Minuten am Tag die Bildschirme schwarz werden lassen. Die haben sich da einfach Rechte herausgenommen. Diesen Einschnitt ins Privatleben sollte es nicht geben, auch wenn der andere vielleicht etwas gemacht hat, das illegal ist, aber das kann man nicht durch eine andere illegale Handlung wieder ausgleichen. Das sind schon Themen, die in der Gesellschaft auch diskutiert werden sollten und im Bildungssystem. Warum sollte man jetzt ein Betriebssystem nehmen, das genauso instabil läuft, das Geld kostet, das einen

neuen Rechner braucht usw. es sind solche Überlegungen. Wenn wir es von der Informatik auf alltägliche Gegenstände oder Betätigungsfelder umlegen würden, dann würde jeder Mensch mit einem Golf fahren und am gleichen Sessel sitzen. Die Offenheit, die sonst überall selbstverständlich ist, dass es Alternativen gibt, wo ich selbst entscheiden kann, das ist in der Informatik einfach noch nicht ausgeprägt. Das hat mit Informationsmangel zu tun und wenn man hier versucht die Angst zu nehmen, dann sollte die Offenheit zumindest in grundlegenden Punkten schon entwickelt werden. Das ist sicher ganz was Wichtiges.

5) Offener oder geschlossener Quellcode? Welche Bedeutung hat dies Ihrer Meinung nach für die Sicherheit einer Software?

Da sind wir wieder bei der Community, die vielleicht auf gewisse Probleme sehr flott reagieren kann, wobei das in einer kleinen Expertenrunde schwieriger sein kann. Ich glaube, obwohl es vielleicht im ersten Moment etwas seltsam klingt, dass freie Software, wo jeder eigentlich in den Code hineinschauen kann, trotzdem sicherer ist, weil es insgesamt noch mehr Gute gibt als Böse. Ich glaube, durch diese natürliche Regelung ist es kein Problem, dass auch die Bösewichte darauf Zugriff haben. Also ich glaube, dass freie Software - das kann man natürlich nicht für jede freie Software punktuell sagen - insgesamt die Möglichkeit hat, sicherer zu sein.

6) Warum glauben Sie, wird in Schulen nach wie vor in so hohem Maße auf proprietäre Software gesetzt?

Das Problem ist, dass Linux erst seit einigen Jahren überhaupt massentauglich geworden ist. Es ist ja heute noch so, dass gewisse Linux Distributionen sehr kantig sind, zumindest von der Software Auswahl und am Desktopbereich auch sehr weit fortgeschritten sind, aber ich würde selber viele davon noch nicht unbedingt empfehlen. Es muss zumindest etwas gleich schauen und die Usability dementsprechend irgendwo auch passen; einfach ein guter Kompromiss sein und erst seit dem Ubuntu-einstieg in die Szene, zumindest ein Jahr danach, kommt es überhaupt für den normalen Konsumenten in Frage.

Das ist zumindest mein Eindruck. Da hat sich einiges getan, das ist nicht nur das Technische, sondern man muss sich auch wohl fühlen, wenn man damit arbeitet; der Faktor gehört da dazu. Ist natürlich jetzt auch leichter, weil man es langsam an die Mac Schiene anpasst und wie Mark Shuttelworth gesagt hat; "Linux muss schicker sein als Mac in Zukunft“. Es muss die sicherste, stabilste und coolste zu bedienende Software werden. Ein Standard-Gnome, wie er immer noch unter Ubuntu mitgeliefert wird, ist einfach zu kantig. Da sind einige Menüstrukturen, die passen, finde ich, noch nicht ganz, aber es sind andere Sachen im Menü, die schauen schon ganz rund aus. Intuitive Handhabung des Systems ist in den letzten Jahren einfach etwas zu kurz gekommen, da ist noch ein Aufholbedarf; das wurde mittlerweile erkannt, wo noch viel passieren kann. Da ist KDE in manchen Geschichten vielleicht weiter, aber der alte Standard KDE war in machen Belangen furchtbar, zwar stark konfigurierbar, aber ich muss dem normalen User etwas hinstellen, wo er im Prinzip nichts machen muss. Das sind sicher solche Entwicklungen, die benötigt werden. Das Problem ist, dass die AHS', die hier sicher flexibler wären, auch weniger Unterstützer bzw. Fürsprecher im Ministerium haben und die HAKs und HTLs sehr laut dagegen gerufen haben. Sie haben argumentiert, Microsoft wird nachher verlangt, und sie haben gemeint, das ist derartig wichtig, dass das Andere nicht Sinn macht usw. und sie haben Leute im Ministerium gehabt, die eingelenkt haben. Mich würde an dieser Stelle ja interessieren, was die Kollegen sagen, wenn man ihnen jetzt das Office 2007 präsentiert, wo ich aus eigener Erfahrung weiß, dass sie da bei OpenOffice gesagt haben: "Das ist ja eh das gleiche, wo ist das Problem?". Selbst unsere Lehrer, die sehr wenig mit dem Computer zu tun haben. Dass jetzt die Lizenzen verlängert worden sind, ist sicher auch nicht förderlich. Aber insgesamt, vor allem beim Kostenfaktor, da könnte man sich schon immense Gelder ersparen, die man dann, bezogen auf Hardware, vielleicht ein bisschen besser investieren könnte, in gescheite Beamer, Aktive-Boards, usw. Wir haben zum Beispiel nicht einmal mehr fixe Budgets für die EDV-Saal-Sanierung, das ist gestrichen. Wenn also irgendwann mal wieder was kommt, dann müssen wir zuschlagen, aber etwas Fixes gibt es nicht mehr. Früher war das so, etwa alle vier fünf Jahre ist wieder ein kleiner Brocken frei geworden, da hat jede Schule etwas bekommen. Die IT Manager werden ja auch grundsätzlich viel zu wenig abgegolten für den Einsatzbereich. Es ist ja nicht so, dass man nur schulische Dinge betreiben muss, sondern auch die Kollegenschaft intensiv mitbetreuen muss - auch in anderen Unterrichtsfächern. Es kommen Computergestützte Klassen, elektronische Klassenbücher ... es läuft immer auf die selben Leute zusammen.

Da könnte man gleich einen Informatiker nehmen und lasst ihn etwas unterstützend eingreifen. Man muss schauen, dass der IT Manager nicht die ganze Arbeit hat für genau das gleiche Salär wie ein Religions- oder Geografielehrer oder sonst irgendeiner. Ich unterrichte jetzt zwar ein bisschen weniger, aber ich arbeite auch dementsprechend mehr als ich unterrichte. Die letzte EDV Umstellung, das war zum Beispiel ein Samstag von 9 - 19 Uhr, das fällt natürlich auch rein.

7) Was spricht Ihrer Ansicht nach für einen Umstieg auf freie Software?

Wie wir schon weitgehend besprochen haben ... wichtig ist einfach, dass die Sichtweise weg von der Schule geht hin zur Schulgemeinschaft: Lehrer, Schüler, Eltern, das ist einfach der wichtige Punkt. Der muss sich generell einfach noch verbessern und dann stellt sich die Frage gar nicht, mit welchem System ich fahre. Ich finde, das beantwortet sich dann eigentlich selbst. Auch wenn es bei allen möglichen Systemen immer irgendwo Schwachpunkte gibt, für mich ist die Entscheidung, wenn ich das berücksichtige, einfach sehr klar; nicht nur in der Schule, weil man mit freier Sofware alle erreicht und eine Möglichkeit bietet, wo man alles legal und kostengünstig machen kann.

8) Mit welchen Schwierigkeiten hat man bei einer Migration zu rechnen?
(Verwaltung, Lehrende oder Lernende)

Der wesentliche Punkt ist, die Unterstützung der Kollegenschaft zu kriegen und da muss man rechtzeitig anfangen, dann ist das kein Problem. Aber man darf diesen Punkt nicht unterschätzen, das ist mindestens genauso wichtig in die ganze Planung einzukalkulieren wie das Technische. Ein System, das braucht immer Wartungsaufwand, ganz egal, ob das ein Windowssystem ist oder ein Linuxsystem, wartungsfrei wird es nicht gehen. Wir haben jetzt die Möglichkeit, es wartungsarm zu machen mit den ganzen Linux USB-Stick Varianten und Überlegungen, was mit einem Windowssystem nicht möglich wäre. Wir haben keine ernsthaften Probleme bei der Migration gehabt. Auf die Lehrer darf man nicht vergessen, das ist ganz wichtig, auch die Eltern darf man nicht vergessen. Wir haben nachher auch Eltern informiert, zuerst bei Elternabenden und im Elternverein, also war auch da die Rückendeckung da. Und technisch: Nicht zu spät anfangen, schauen, dass man auch die Ferien zum Testen hat, wo man letzte Troubles noch bereinigen kann.

Natürlich ist es positiv, wenn in der Schule mehr als eine Person dafür verantwortlich ist; wenn sich mindestens zwei Leute finden, die dann auch Engagement zeigen, gewisse Sachen zu lösen, das wäre von Vorteil.

9) Ist die Möglichkeit, die Funktionsweise eines Programms zu verändern, relevant für Bildungseinrichtungen? Informatikunterricht? Warum? Auf welche Weise?

Vielleicht weniger bei der typischen Standardsoftware, aber es gibt Moodle, das kann man sehr stark anpassen oder Wikis oder irgendeinen Blog. Wir haben natürlich auch das Betriebssystem angepasst, aber hier geht es wohl auch um die Betrachtung, was hat der Normaluser jetzt davon, der wird natürlich das Betriebssystem selber nicht anpassen, aber er profitiert dann wieder von jenen, die es können, weil es irgendwer gekonnt hat und auch die Erlaubnis hatte es zu machen. Grundsätzlich - um in der Schule zu bleiben und die Wartung einmal kurz hernehmen - man kann sich geschickt ein Skript schreiben, um die Wartung durchzuführen; man kann gewisse Fehler, die man kennt, ausbessern, wenn man das Know-how hat; man kann ins System eingreifen. Aber wie gesagt, der Schüler selbst hat gewisse Anwendungen wie solche, die im Web2.0 beherbergt sind zum Beispiel, wo auch da eigentlich der Gedanke aufkommen könnte, dass man was ändern kann.
Mit den Schülern arbeiten wir natürlich nicht direkt am Quellcode, also OpenOffice Fehler aus dem Quellcode heraussuchen, das machen wir nicht. Das ist auch immer eine Ressourcenfrage.

10) Könnten Sie in wenigen Worten erläutern, warum offene Schnittstellen und Standards von Bedeutung sind? (in inwiefern ist das für eine Schule wichtig?)

Man kann das vielleicht gut an einem Beispiel erörtern: Internettelefonie VoIP: Man hat das Programm Skype zum Beispiel, das ist sehr bekannt. Viele setzen es wahrscheinlich auch ein, aber man kann eigentlich nur von einem Skype-Client zum nächsten Skype-Client telefonieren. So stellt sich jetzt die Frage, warum kommuniziert das Ding nicht mit anderen Programmen? Wir sind da wieder nur auf ein Tool beschränkt, wobei es verschiedene Werkzeuge gibt, mit denen man telefonieren kann.

Da ist irgendwo der Punkt, dass es eigentlich nicht flexibel ist, was jetzt nicht unbedingt für den Einzelnutzer Sinn macht, weil er muss dann etwas verwenden, was ein anderer vorgibt und kann keine Alternativen in Betracht ziehen. Ich glaube, bei Skype sieht man es sehr schön, dass es nicht mit SIP konform geht. Das ist ein Standard für alle VoIP Clients, die sich daran halten könnten und das Protokoll nutzen könnten. Oder bei Office: Wenn man Dateiformate von Office anschaut, Microsoft Office z.B., das sind ja Sachen, da weiß keine Person wirklich, wie das aufgebaut ist. Man hat im OpenOffice die Möglichkeit zu importieren, was inzwischen meistens gut klappt, aber auch noch nicht immer; Formatierungen verhauen es dann oft. Das sind so Sachen, wo man einen Schritt weiter gehen müsste. Behörden müssen ja auch irgendwo Dokumente langzeitfähig speichern; das mit einem geschlossenen Standard zu machen ist nicht sinnvoll, wo nach „zig“ Jahren nicht mehr garantiert werden kann, dass man die Datei dann noch aufmachen kann, weil man nicht weiß, wie das Dokumentenformat geschrieben worden ist; weil man ja nicht hineinschauen kann, was z.B. bei ODF, dem offenen Dokumentenformat, problemlos möglich wäre. Es wird mittlerweile nun auch EU-weit gefordert, dass Behörden offene Standards einsetzen müssen, wenn es keine anderen Produkte gibt, die einen immensen Vorteil hätten, den das entsprechende freie Produkt ihnen nicht bieten würde. Als Schule profitiert man natürlich indirekt oft von gewissen Strömungen. Das Office-Problem hat man natürlich genauso in der Schule, z.B. bei der Umstellung von Office 2003 zu 2007. Manche haben das 2007er schon und kommen mit ihren Referaten in die Schule; das ist zum Beispiel ein DOCX und dann machen sie große Augen, weil man es nicht öffnen kann. Warum? Weil es einfach kein offener Standard ist. Wäre es ein offener Standard, hätte OpenOffice es schon längst integriert. Es ist zwar ISO zertifiziert, aber es ist nicht wirklich offen, es entspricht einfach nicht den Kriterien eines offenen Standards. Wir versuchen jetzt die Schüler natürlich nicht dahin zu drängen, das Linux zuhause zu nutzen. Aber dass sie die Software, die wir unter Linux verwenden, zumindest zuhause installieren, da schauen wir schon, dass sie das machen. In meinen Klassen bekomme ich, wenn ich jetzt Textverarbeitung hernehme, fast alles in ODT. DOC akzeptiere ich, keine Frage, das kann ich auch aufmachen; DOCX z.B. akzeptiere ich derzeit nicht. Die Schüler sollen auch lernen, das Problem zu sehen, das durch solche Inkompatibilitäten einfach existiert.

11) Ist die illegale Vervielfältigung von Software ein Problem, mit dem sich Schulen auseinandersetzen müssen?

In der Schule gibt es natürlich keine illegale Vervielfältigung, die hat es aber vor der Linux Umstellung auch nicht wirklich gegeben. Sicher kommt es mal vor, dass irgendwer eine CD brennt oder so, aber das kann ich jetzt nicht verifizieren, das ist auch gar nicht mein Job, jetzt überall mit dem erhobenen Zeigefinger herumzugehen. Ich denke auch, wenn ich ihnen jetzt etwas gebe, womit sie legal arbeiten können, dass sich die Frage gar nicht stellt. Bei Schulen, die proprietäre Software einsetzen, ist es klar, die Schulen haben es natürlich legal. Aber bei den Schülern, die das dann zuhause einsetzen, ist es sicher zu einem großen Anteil nicht legal. Ich bin überzeugt, dass das ein sehr hoher Prozentsatz sein wird. Das ist natürlich Privatsache, aber kann man das so abkapseln? Die Schule von zu Hause? Ich kann ja nicht sagen, Bildbearbeitung, kauft euch Adobe Photoshop und zahlt ein paar hundert Euro, damit ihr zuhause üben könnt, das wäre natürlich der falsche Weg. Die Frage ist einfach, wo profitieren alle am meisten. Mit Gimp kann ich alles, was ich in der Schule brauche, perfekt umsetzen und ich hab auch schon Vergleichskurse gesehen, wo das derartig ähnlich ist; wo die Menüpunkte sogar gleich heißen, weil man davon nicht weggehen kann.
Klonen oder freistellen von einem Objekt funktioniert da gleich wie dort. Grundsätzlich... die Schule könnte natürlich sagen, uns interessiert als Schule nur, was im Haus passiert. Im Grunde genommen sehe ich das nicht als richtig an, finde ich das bedenklich, wenn das der Fall wäre oder der Fall ist. Vor allem, wenn ich mir denke, dass die Anforderungen auch von finanzieller Seite an die Eltern sicher nicht rückläufig sind, sondern im Bildungssystem zunehmen; in 10-15 Jahren wird vielleicht jeder im Regelunterricht, nicht im Informatikunterricht, mit einem Netbook in der Schule sitzen, das nicht die Schule zahlt, sondern die Eltern. Es wird für sie nicht billiger, daher ist es schon eine Selbstverständlichkeit, dass ich nicht leichtfertig eine Lösung vorschlage, die einfach nicht die beste Lösung ist und finanzielle Belastungen schafft, die einfach nicht notwendig sind.

12) Wie sollte, im schulischen Kontext, die Software-spezifische Ausbildung junger Menschen gestaltet sein?

Die Schüler sollen erkennen, was ist z.B. eine Zeichenformatierung, was ist z.B. eine Absatzformatierung; dass das eine z.B. nur eine Zeichenformatierung sein kann oder umgekehrt. Dann ist ja oft schon nahe liegend, wo ich das finde. Ich versuche sie wirklich vom blinden Hinklicken weg zu bringen.
Wenn man versucht das zu berücksichtigen, dann fällt auch ein Umstieg nicht schwer. Ich verwende daheim auch kein Windows, finde aber in einem Outlook-Mailclient auch meine Sachen, da muss ich halt kurz mal suchen. Bei E-mail z.B. lehren wir, was braucht man überhaupt um das zu ermöglichen, um einen Mailclient am Rechner zu installieren, welcher Client auch immer das ist, völlig egal. Es ist einmal wichtig, dass ich weiß, was für Daten ich brauche. Username, Passwort, POP oder IMAP, dass man überhaupt eine Vorstellung davon hat, dass irgendwas dafür zuständig ist, das Mail überhaupt hineinzubringen und etwas anderes, um es herauszubringen; das müssen die Schüler wissen.

13) Kann es die Aufgabe einer Schule sein, wirtschaftlichen Monopolen entgegen zu wirken?

Also grundsätzlich ist es sicher nicht die Aufgabe einer einzelnen Schule, gegen irgendwelche wettbewerbsverzerrenden Verhältnisse in einer globalen Marktwirtschaft zu agieren; das kann nicht der Sinn sein. Aber die Schule hat sehr wohl den Auftrag, für ihren Fall zu entscheiden, was das Beste ist und das ist einfach aus den Konsequenzen, die wir vorher schon diskutiert haben, in vielen Fällen sicher kein proprietäres Produkt. Also nicht direkt, so dass man jetzt versucht, eine Quasi-Monopolstellung zu Fall zu bringen, sondern einfach wieder eine Kosten/Nutzenrechnung, die von Fall zu Fall einfach anzustellen ist. Man sollte sich immer dem jeweiligen Fall anpassen, einfach immer schauen, was braucht man eigentlich, was ist dafür die beste Lösung. Und indirekt läuft es dann vielleicht wieder zusammen, dass das Eine das Andere impliziert, aber das Ziel selbst kann es nicht sein, höchstens eine Folgewirkung dessen, was wirklich wichtig ist. Das ist einfach eine andere Sichtweise.

14) Gibt es etwas das wird noch nicht besprochen haben, zu diesem Thema aber noch gesagt werden sollte?

Ich glaube, dass wir als Informatiklehrer eine Riesenaufgabe haben, mit Weitblick zu agieren, vorurteilsfrei zu agieren und in diesem schnelllebigen Bereich auch die Fähigkeit nutzen, auch einmal andere Software anzuschauen und vorausschauenden, verantwortungsvollen Umgang damit zu machen.

8.2.3 MMag. Marco Delbello

Steckbrief:
Studium Physik und Sport (Lehramt)
Studium Sportwissenschaften
Ausbildung zum Informatiklehrer für Hauptschulen
Laufendes Studium der Informatik
Lehrer am BG /BRG Ingeborg Bachmann Klagenfurt

1) Setzen Sie freie Software privat ein?

Ja.. kurz und bündig. Ich hab schon mehrmals versucht, auf ein freies Betriebssystem umzusteigen. Das war das erste mal das SuSE Linux, da hab ich mir von Kofler dieses Buch gekauft für Linux, angelehnt an SuSE. Da hab ich probiert, damals schon umzusteigen, weil ich mich über das Windows geärgert habe und jetzt habe ich ein Dualbootsystem. Ich habe ein Notebook, und seit ich das Ubuntu gefunden habe, bin ich sehr zufrieden damit, weil es einfach sehr viele Treiberprobleme, die man mit Hardware hatte, in den Griff bekommen hat. Es ist nicht mehr so zeitaufwendig wie vorher. Ich habe mich anfangs mit dem SuSE sehr geplagt und bin dann wieder zurück zu Windows und dann, als ich das Ubuntu entdeckt habe, bin ich dabei geblieben. Eben habe ich ein neues Notebook gekauft mit einer Stiftfunktion. Wenn man weiß, was man einstellen muss, klappt das auch mit dem Stift.. das ist ein Wacom-Device.. das geht nur mit dem Stift. Der Vorteil davon ist natürlich klar, weil wenn ich die Hand ablege, dann würde der Cursor immer hin und her springen. Jedenfalls bin ich jetzt wieder viel auf Vista, weil das neue Office Paket gekommen ist und bei Powerpoint Präsentationen kann ich mit dem Stift einfach so drauflos schreiben. Wenn ich eine PP Präsentation habe, und ich habe da meine Folien, habe ich unten einen Punkt, wo ich drauf klick und dann kann ich zu den Folien etwas dazu schreiben. Das ist momentan der einzige Grund, warum ich im Vista unterwegs bin und ich muss sagen: Abgesehen davon, dass es extrem viel Arbeitsspeicher braucht, ist es gar nicht so übel.. relativ stabil.. am Anfang hatte ich ein paar Bluescreens in Verbindung mit Firefox, aber sonst passt es.

2) Benötigen Sie Anwendungssoftware, die nicht oder nur unzureichend durch das Angebot freier Software gedeckt würde?

Also ich muss sagen; in meinem Unterricht nicht. Ich weiß, dass Kollegen, die nicht Informatiker sind und dennoch Informatik unterrichten, Probleme haben sich auf eine andere Software umzustellen. Das heißt, sie arbeiten mit einer speziellen Software und nur mit der und sie gehen dann zum Administrator und sagen, ich möchte genau die Software installiert haben und mit der arbeiten. Auch in anderen Fächern werden die PCs häufig verwendet. Nur zur Videoverarbeitung habe ich einmal ein kommerzielles Produkt verwendet. Das neue Office Paket ist auch sehr übersichtlich gemacht, das neue Userinterface, also ich hoffe, dass da OpenOffice nachzieht. Im Konferenzzimmer hieß es aber auch zu Office 2007: Nein, das ist wieder so eine Umstellung usw. Auf der pädagogischen Hochschule haben sie am Anfang eine Veranstaltung gehabt, wo es um Vista gegangen ist, um Vista und Office. Da haben sie einen Workshop angeboten, also Fortbildung, aber das war mehr eine Verkaufsveranstaltung als was anderes, das war furchtbar. Die Verwaltung erfordert allerdings spezielle Software. Die Verwaltung und das pädagogische Netz sind komplett getrennt voneinander in der Schule und in der Verwaltung kümmert sich der Landesschulrat um die ganzen Lizenzen für die Software, das ist zumindest bei uns in Kärnten auch allgemein so üblich. Wenn da was kaputt ist, kommt auch wer vom Landesschulrat. Vorher war das in einem, aber seit einem oder zwei Jahren ist das so, dass das vom Landesschulrat teilweise auch fern gewartet wird.

3) Was halten Sie von diesem Statement?

Ich bin natürlich dafür, dass Software weitergegeben werden kann; weil das ist in meinem Interesse, wenn ich mit den Schülern irgendetwas Neues am Computer mache, dass ich ihnen dann auch die Software mit nach Hause geben kann, damit sie das zuhause üben können. Das ist eben in meinem Interesse, deswegen bin ich dafür, dass es weiter gegeben werden kann. Zum Quellcode selber, auch das ist sicher von Vorteil, weil das Programm ja verbessert werden kann durch eine große Gemeinschaft, die sich dafür interessiert. Das ist natürlich von Vorteil, wenn ich ein Problem habe, wie kürzlich mit meiner Puls-Uhr, dann bekommt man auch tolle Unterstützung, und das ist natürlich auch vom Sozialen her eine tolle Sache.

4) Sollten und können Überlegungen dieser Art bei der Wahl der Software für eine Schule von Bedeutung sein?

Ich denke schon, dass es für eine Schule spricht, wenn sie sich mit freier Software auseinandersetzt und die auch im Unterricht einführt, weil in der Schule sollen ja auch genau diese sozialen Sachen vermittelt werden. Das ist ein zusätzlicher Punkt, wo man mit der Software zeigen kann, schaut her, bei der Software ist das auch so. Es gehört in gewisser Weise zur Idee einer allgemeinbildenden Schule, auch solche Werte zu vermitteln und mit gutem Beispiel voranzugehen.

5) Offener oder geschlossener Quellcode? Welche Bedeutung hat dies Ihrer Meinung nach für die Sicherheit einer Software?

Zwei Seiten: Auf der einen Seite, wenn ich sage, der Code ist offen, kann es sein dass hier jemand etwas einbaut, es wieder ins Internet stellt und ich das dann herunterlade und dadurch ein Sicherheitsproblem bekomme, ohne es überhaupt zu wissen; das ist sicher ein gewisses Risiko. Wenn also irgendwer nach einer Software sucht und erwischt dann diese veränderte Version und schaut nicht genau, startet er diese bei sich und hat dadurch ein Sicherheitsproblem. Man muss also letztlich auch auf die Paketquellen vertrauen, auch wenn man nur die offiziellen von Ubuntu z.B. verwendet. Der Vorteil ist natürlich, je mehr Leute sich den Code ansehen, desto sicherer wird auch die Software.

6) Warum glauben Sie wird in Schulen nach wie vor in so hohem Maße auf proprietäre Software gesetzt?

Ich denke, dass vor dem Microsoft Agreement... also zuerst war es ja so, dass jede Schule sich praktisch selber um Lizenzen hat kümmern müssen; das waren hauptsächlich Microsoftprodukte. Das heißt, wäre eine Kontrolle gekommen, hätte man für jeden Computer eine Lizenz vorweisen müssen. Ich glaube, das war so um 2003, nachdem viele Schulen gesagt haben, sie möchten nicht mehr in diesem halb legalen Bereich unterwegs sein.

Weil sie natürlich nicht so viel Geld zur Verfügung gehabt haben, um so viele Lizenzen zu kaufen - es kann ja nicht das ganze Geld von der Schule in den Informatikbereich fließen - war damals schon die Tendenz, auf Open Source Software umzusteigen, auf freie Betriebssysteme. Die Schulen haben begonnen umzusteigen; dann hat Microsoft reagiert und hat eben dieses Agreement mit dem Bundesministerium getroffen und hat praktisch bundesweit Lizenzen angekauft für alle Schulen und die Schulen haben sich selber nicht mehr kümmern müssen um die Lizenzen. Sie haben praktisch einen Key bekommen und das war es. Ich hab sowieso nie verstanden, wieso Microsoft an eine Ausbildungsstätte das Produkt nicht einfach gratis vergibt, weil es werden dann ja alle darauf ausgebildet und die kaufen das Produkt für daheim und später auch wieder. Aber durch dieses Agreement haben sie natürlich den Fuß in der Tür. Vor einem Jahr ist das ausgelaufen und wurde nun wieder verlängert. Das Agreement ist also auf der einen Seite dafür verantwortlich und auf der anderen einfach die Unflexibilität der Lehrer, vom Bekannten zu irgendetwas Neuem zu gehen. Ich würde das natürlich sofort machen, dass ich die Kollegen auf die neuen Produkte einschule und auf die OSS einführe, aber ich glaube, es würde auf sehr starken Widerstand stoßen.

7) Was spricht Ihrer Ansicht nach für einen Umstieg auf freie Software?

Für einen Umstieg steht in erster Linie die Administration der Software, aus Sicht des Computeradministrators, weil ich sehe, wie der sich oft plagt, wenn er irgendwelche Sachen bei Windowssystemen wiederherstellen muss, weil das alles so ein starres System ist; man kommt kaum dazu, einzelne Sachen wiederherzustellen. Vorhin hab ich den Schülern gesagt, sie können auf Ubuntu alles ausprobieren und ich hab das dann wieder zurückgesetzt, ganz einfach über das Gconf-Tool. Ich hab gesagt, sie können Sachen hinzufügen, irgendwelche Icons und sonst was kaputt machen und mit einem kleinen Konsolenbefehl kann ich das alles wieder zurücksetzen. Und überhaupt die Absicherung von Systemen: Ich kann unter Linux unglaublich einfach Bereiche absichern mit der Rechtevergabe usw. Ich tue mir da wesentlich leichter als bei Windows, da könnte man natürlich auch restriktiv vorgehen und alles sperren, aber dann können sie z.B. nicht einmal - in der Schule, damit sie eine sichere Verbindung aufbauen können, muss der Proxy im Browser eingegeben werden – wenn ich das also zumache, dann können sie nicht einmal das machen.

Auf Linux kann sich der User frei bewegen, dass ist ja sein Account. Natürlich ist das bei uns nicht optimal gelöst, momentan hat jede Maschine einen eigenen Useraccount. Diese Idee mit dem USB-Stick der Kremser Schule ist da natürlich eine tolle Sache. Und auch, dass man den Schülern die Software mitgeben kann. Man will natürlich mit gutem Beispiel vorangehen; den Schülern gecrackte Software weitergeben kann man natürlich nicht machen, wobei dies bei den Schülern Alltag ist. Wenn ich ihnen sage, wir arbeiten mit dieser OSS in Windows z.B. mit dem GIMP und ich darüber hinaus erkläre, dass wir mit Gimp und nicht mit Photoshop arbeiten, weil ich ihnen das weitergeben kann, dann kommt natürlich sofort als Antwort, das ist kein Problem, das findet man ja im Internet und den Key auch usw. und dann muss natürlich von der Lehrerseite kommen, dass das nicht geht. GIMP ist z.B. ein Spitzenbildbearbeitungsprogramm und wenn die Lehrer mit dem angefangen hätten, würden alle mit dem arbeiten.

8) Mit welchen Schwierigkeiten hat man bei einer Migration zu rechnen? (Verwaltung, Lehrende oder Lernende)

Die Hauptschwierigkeit ist sicher die Akzeptanz der Kollegen. Auf jeden Fall muss man als erstes die Kollegen für sich gewinnen und wirklich gute Gründe für den Umstieg nennen.

9) Ist die Möglichkeit, die Funktionsweise eines Programms zu verändern, relevant für Bildungseinrichtungen? Informatikunterricht? Warum? Auf welche Weise?

Eher weniger, würde ich sagen, wenn sich jemand wirklich gut auskennt, kann es im Verwaltungsbereich sinnvoll sein, ein Programm an die Bedürfnisse der Schule anzupassen. Aber wie gesagt... eher im Verwaltungsbereich. Auch das von Lehrern machen zu lassen würde wohl den Zeitrahmen sprengen. Im Unterricht hab ich das auch noch nie gemacht, ich wüsste auch nicht, wie ich das machen soll. Ich denke dass das für eine allgemeinbildende Schule zu speziell ist.

10) Könnten Sie in wenigen Worten erläutern, warum offene Schnittstellen und Standards von Bedeutung sind? (in inwiefern ist das für eine Schule wichtig?)

Wenn ich in der Lage bin, Sachen dazu zu programmieren bzw. verschiedene Programme miteinander zu verbinden, ist das ganz sicher von Vorteil, wenn diese Schnittstellen offen sind. Hausübungen und Referate werden teilweise in der Schule digital abgegeben, die kommen dann auf .doc. Aber wenn dann .docx herangetragen werden, können wir es auch öffnen, weil wir ja die neue Office Suite am Laufen haben. Ich habe z.B. jetzt das Problem gehabt, dass ich in einem Programm E-mail Konten konfiguriert habe und meine E-mails dort abgelegt habe. Ich hab dann ein anderes Programm gefunden und möchte die Emails von einem Programm in das andere übertragen, da hab ich bis jetzt immer Probleme gehabt. Es gibt z.B. auch für Konten keine Exportiermöglichkeit. In die eine Richtung ist das meist kein so großes Problem, also ich kann auch Word-Dokumente in OpenOffice öffnen, nur umgekehrt halt nicht.

11) Ist die illegale Vervielfältigung von Software ein Problem, mit dem sich Schulen auseinandersetzen müssen?

Es ist niemand da der kontrolliert, ob in der Schule auf den Schulcomputern illegale Software verwendet wird oder nicht. Ich glaube, wenn es jemanden geben würde, der das kontrolliert und auch bei den Leuten zuhause kontrollieren würde, wenn hier restriktiv durchgegriffen werden würde, dann würden viel mehr Leute auf OSS umsteigen. Die einzigen, die kontrolliert werden, sind Firmen und die haben auch in der Regel Lizenzen. Es ist klar, von der Schule wird keine Software an die Schüler weitergegeben. Daher versuche ich auch freie Software zu verwenden, damit ich das auch weitergeben kann. Ich kann doch auch von den Schülern nicht verlangen, übt das zuhause, ohne ihnen die Software zur Verfügung zu stellen. Sonst zwinge ich sie damit ja, irgendwas zu kaufen, Geld auszugeben, was die wenigsten machen würden und in Folge würden sie es sich illegal besorgen.

12) Wie sollte, im schulischen Kontext, die Software-spezifische Ausbildung junger Menschen gestaltet sein?

Ich finde es wichtig, keine Programmschulung zu machen, sondern Methoden zu lehren, wie sie sich selber zurechtfinden. Man kann da jetzt mehrere verschiedene Bildbearbeitungsprogramme verwenden z.B. um auf die kleinen Unterschiede aufmerksam zu machen. Und es kommt, glaub ich, auch darauf an, wie man es unterrichtet, wenn man es rein von der Bezeichnung her so verwendet, wie es im Programm benannt ist oder ob man es eher allgemein hält.. wir unterrichten also nicht Word oder Excel, sondern Textverarbeitung und Tabellenkalkulation. Ich schau halt, dass sie ein möglichst breites Spektrum über alle Sachen, die man mit dem Computer machen kann bekommen und überall ein bisschen hineinschnuppern. Wenn sie sich dann für ein Thema mehr interessieren, können sie sich eigentlich selber vertiefen. In der fünften Klasse gibt es einen Überblick und im Wahlpflichtfach wird dann vertieft. Es ist, glaub ich, auch ein wichtiger Schritt, dass man ihnen die Angst nimmt vor einer neuen Software. Dass sie sagen, ich kenne nur das und nehme daher nur das, ist bei den Schülern nicht so das Problem; das fällt vielleicht eher bei den Kollegen auf, dass sie da sehr resistent sind.

13) Kann es die Aufgabe einer Schule sein, wirtschaftlichen Monopolen entgegen zu wirken?

Ja. Entgegenwirken kann ich, indem ich in der Schule einfach freie Software verwende. Wenn das wirklich flächendeckend in ganz Österreich passiert, gehen die Kinder dann raus, kennen sich mit dem Linux Betriebssystem aus und werden dieses zuhause auch verwenden. Der Zuständigkeitsbereich einer Schule wird das höchstens indirekt, da es passiert, dass die Software, die wir verwenden, später auch eingesetzt wird. Ich glaube, die Verbreitung von Windows war sicher möglich, da es absolut keinen Kopierschutz hatte und somit konnte es sich im Privatbereich verbreiten.

14) Gibt es etwas das wird noch nicht besprochen haben, zu diesem Thema aber noch gesagt werden sollte?

Eigentlich haben wir das recht ausführlich besprochen. Fällt mir noch was ein? Ich denke auf die Schnelle nicht...

9 Quellenverzeichnis

Eine vollständige Sammlung sämtlicher Internetquellen die der Vertiefung in dieses Thema dienten findet sich unter: *http://foss.xapient.net*

ABOCK: Bockover, Aron:
Banshee 1.4 hits the streets, packed with Awesome
17. November 2008
http://abock.org/2008/11/17/banshee-14-hits-the-streets-packed-with-awesome/

BECTA: British Educational Communications and Technology Agency:
Open source software in schools: a study of the spectrum of use and related ICT infrastructure costs
BECTA, Mai 2005
http://publications.becta.org.uk/

BILL: Gates, Bill:
An Open Letter to Hobbyists
3. Februar 1976
http://www.microsoft.com/about/companyinformation/timeline/timeline/docs/di_Hobbyists.doc

BING: Bingel, Peter:
Von Computern und Menschen - Linux im Einsatz an einer Hauptschule
2003
http://www.bpb.de/

BMI: Bundesministerium des Innern (Deutschland):
Migrationsleitfaden, Version 3.0, Erste Auflage
Referat IT 2, April 2008

BSI: Bundesamt für Sicherheit in der Informationstechnik (Deutschland):
Strategische Position des BSI zu Freier Software
Dezember 2008
http://www.bsi.bund.de/oss

FLOSSI: Ghosh, Rishab Aiyer:
Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU
Europäische Kommission, 20. November 2006

FSFE: Free Software Foundation Europe:
Was ist freie Software
FSFE, 2008
http://www.fsfeurope.org/documents/freesoftware.de.html
Helfen Sie mit!
FSFE, 2008
http://www.fsfeurope.org/documents/printable/folder.de.0.pdf

FSF: Free Software Foundation:
High Priority Free Software Projects
18. November 2008
http://www.fsf.org/campaigns/priority.html

FSPIR: Jakobs, Joachim:
Freie Software löst das Problem illegaler Raubkopien
Interview im Pressetext, 21. Februar 2006
http://www.pressetext.at/pte.mc?pte=060221020

GRASS: Grassmuck, Volker:
Freie Software Zwischen Privat- und Gemeineigentum
Bundeszentrale für politische Bildung Bonn 2004, 2. aktualisierte Auflage

GAGE: Stieler, Wolfgang:
Alles wird als Open Source offengelegt
Interview mit John Gage, 13. Oktober 2006
http://www.heise.de/tr/Alles-wird-als-Open-Source-offengelegt--/artikel/79239/

HMIOS: Lettice,John:
How Microsoft invented open source, by Billg
9. November 2001
http://www.theregister.co.uk/2001/11/09/how_microsoft_invented_open_source/

IDABC: Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens:
European Interoperability Framework for Pan-European eGovernment Services
European Communities, 2004,
http://europa.eu.int/idabc/

JFF: Torvalds, Linus, Diamond David:
Just for fun : wie ein Freak die Computerwelt revolutionierte
Aus dem Amerikan. von Doris Märtin, München : Dt. Taschenbuch-Verl. , 2002

KCI: Nüttgens, Markus, Tesei, Enrico:
Open Source – Konzept, Communities und Institutionen
Heft 156, Januar 2000,

MCASE: Contorer, Aaron:
Internal Microsoft memo drafted for Bill Gates, S.126 – S.127
21. Februar 1997
In: The Commission of the European Communities:
EC report on the Microsoft case
European Commission, 21. April 2004
http://europa.eu.int/comm/competition/antitrust/cases/decisions/37792/en.pdf

MENA: Meuser, Michael, Nagel, Ulrike:
ExpertInneninterviews – vielfach erprobt, wenig bedacht. Ein Beitrag zur qualitativen Methodendiskussion, S.442 – S.467
In: Garz, Detlef, Kraimer, Klaus:
Qualitativ Empirische Sozialforschung. Konzepte, Methoden, Analysen
Opladen (Westdeutscher Verlag), 1991

MFOSC: Stoveland, Jan Fredrik:
Managing Firm-Sponsored Open Source Communities - A case study of Novell and the openSUSE project
Master Thesis, Universität Oslo, Mai 2008

MUENCH: Portal München Betriebs-GmbH & Co. KG:
2 Jahre LiMux - Offene Standards, freie Software, starke Wirtschaft
Pressegespräch mit BM Christine Strobl, 9. Juli 2008
http://www.muenchen.de/Rathaus/dir/limux/publ/242906/presseinfo.html

OSI: Coar, Ken:
The Open Source Definition
Open Source Initiative, Juli 2006
http://www.opensource.org/docs/osd

OSOR: Gerloff, Karsten:
Declaration of Independence: The LiMux Project in Munich
Europäischer Open-Source-Informationsdienst OSOR
(Open Source Observatory and Repository), September 2008
http://osor.eu/

OSPP: Perens, Bruce:
Open Standards Principles and Practice
2008
http://perens.com/OpenStandards/Definition.html

QFLT: Diamond, David:
Questions for Linus Torvalds
Interview mit Linus Torvalds, 28. September 2003
http://query.nytimes.com/gst/fullpage.html?
res=9F04E2DF163DF93BA1575AC0A9659C8B63&sec=technology

REVOS: Moore, J.T.S.:
Revolution OS
Dokumentarfilm, Wonderview Productions, 15. Februar 2002

STOSS: Lutz, Brigitte / Potakowskyj, Johann / Richter, Klaus, u.a.:
Studie OSS. Open Source Software am Arbeitsplatz im Magistrat Wien.
Magistrat Wien MA14, November 2004

TCAB: Raymond, Eric Steven, 2000
The Cathedral and the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary
O'Reilly Verlag, 2000

TOP500: Top500.org:
Operating system Family share
Noveber 2008
http://www.top500.org/stats/list/32/osfam

USENET: Torvalds, Linus:
What would you like to see most in minix?
Email, 25. August 1991
http://www.linux.org/people/linus_post.html

WIFS: Pick, Michael:
What ist Free Software
Interview mit Richard Stallman, 24 Oktober 2006
http://www.masternewmedia.org/news/2006/10/24/what_is_free_software_a.htm

XMON: Gueras, Sylvie:
Browsers Barometer
AT Internet/XiTi.com, 16. Oktober 2008
http://www.xitimonitor.com/en-us/browsers-barometer/browsers-barometer-september-2008/index-1-2-3-145.html

10 Abbildungsverzeichnis

Abbildungen deren Herkunft nicht explizit angegeben wurde stehen entweder unter einer freien Lizenz oder wurden vom Autor selbst erstellt.