

Linux in der Verwaltung

Ronny Edler, Bertram Wölk, Radoy Pavlov, Konrad Lorentz

16. Dezember 2004

Zusammenfassung

Verwaltungen, als Institutionen der Staates, bewegen sich, was die Entscheidungsfindung, - insbesondere für IT-Projekte - betrifft, in einem natürlichen Diskursbereich von politischen, sowie von Sicherheits- und Kostenaspekten. Ausgehend von einigen konkreten Projekten werden wir in der vorliegenden Ausarbeitung ebenjene Aspekte, ob des Einsatzes von Linux in der Verwaltung näher untersuchen. Dabei zeigt sich Linux, was die Sicherheit und die Kosten angeht vergleichbaren proprietären Lösungen ebenbürtig. Die Politik betreffend, zeigen sich vor allem in Europa durchaus Tendenzen, den Einsatz über das aktuelle Maß hinaus zu forcieren.

Inhaltsverzeichnis

1	Open Source Software	5
2	Projektvorstellung	6
2.1	Einleitung	6
2.2	Linux in München	6
2.3	Linux in Schwäbisch Hall	7
2.4	Linux in einem mittelständischem Unternehmen	7
3	Open-Source Software und Politik	9
3.1	Die Situation in Europa	9
3.1.1	England	9
3.1.2	Frankreich	9
3.1.3	Spanien	9
3.2	Die Situation in Deutschland	9
3.2.1	Die Bundesregierung	9
3.2.2	Migrationsleitfaden	10
3.2.3	Justitia online	10
3.2.4	IBM/Microsoft Rahmenverträge	10
3.2.5	CDU	11
3.2.6	FDP	11
3.2.7	PDS	11
3.2.8	Treffen in Berlin, 10.11.2004	11
3.3	Fazit	12
4	Der Kostenaspekt	13
4.1	Migration	13
4.1.1	Migrationsgründe	13
4.1.2	Migrationswege	13
4.1.3	Migrationsarten	14
4.2	Kosten	16
4.2.1	Monetäre Analyse	16
4.2.2	Nutzwertanalyse	16
4.2.3	IT-WiBe 21	16
4.2.4	Total cost of ownership	17
4.3	Conclusio	20
5	Sicherheit von Open-Source-Software	21
5.1	Grundlegende Merkmale kommerzieller und freier Softwareentwicklung	21
5.2	Open-Source-Entwicklungsmethoden wirken sich auf die Codequalität aus	21
5.3	Umgang mit Sicherheitslücken	23
5.4	Erleichtert die Veröffentlichung des Quellcode Angreifern Sicherheitslücken zu finden?	23
5.5	Vergleich von Windows- und Linuxdesign	24
5.5.1	Einzel- vs. Mehrnutzersystem	24
5.5.2	Monolithische vs. modulare Architektur	24
5.5.3	Starke vs. schwache Bindung an ein RPC-Modell	25

5.6	Sicherheitsvergleich bei Webservern in der Praxis	26
5.7	Fazit	28

1 Open Source Software

Obwohl heute in aller Munde ist der eigentliche Begriff *Open Source* erst 1998 entstanden. (Dieckheuer und Kooths (2003), S.32) Dieckheuer und Kooths (2003) führt weiter aus, daß die ursprüngliche Idee einer *freien* Software jedoch bis in die 1980er Jahre zurückreicht. Im folgenden wird eine weitere Unterscheidung zwischen *Free* Software und *Open* Software *nicht* vorgenommen. Hierzu sei auf einschlägige Quellen, wie z.B. Free Software Foundation (2004) verwiesen. Wir werden im folgenden nur den Begriff der *Open Source* benutzen.

Nach Leiteritz (2002) ist Open Source Software (OSS) alljene Software, die ihren Nutzern folgende Rechte zusichert:

- Jeder hat das Recht, die Software nach eigenem Ermessen zu nutzen.
- Der Quelltext muss jedem Benutzer offengelegt werden[,] oder es muß auf eine frei zugängliche Stelle verwiesen werden, wo er erhältlich ist.
- Der Benutzer hat das Recht, die Software zu modifizieren und in modifizierter Form weiterzuverteilen.
- Die Lizenz darf niemanden im Verkauf oder der Weitergabe der Software in Form einer Softwarezusammenstellung einschränken.

Wohingegen proprietäre Software alljene ist, die ausschließlich in Binärform vertrieben und deren Weiterverbreitung, Vervielfältigung und Modifizierung untersagt wird (Leiteritz (2002)). (Für eine genauere Unterscheidung siehe auch Open Source Initiative (2004) bzw. Snoopy und Müller (1999) für eine deutsche Übersetzung.)

Oder wie Lessig (2002), S.52f ausdrückt:

„Open Source and free software give consumers and the public something more than proprietary software does: the ability to tinker and modify. Such software gives the public the benefit of the information contained within the code. [...]

Proprietary software is made available upon the payment of a price (which sometimes is zero). In exchange for a price, the user ordinarily licences the object code. Object code [...] does not transmit the information it contains; it is simply a machine that induces another machine to function in a particular way.“

2 Projektvorstellung

2.1 Einleitung

In diesem Teil der Ausarbeitung werden, ausgehend von den Beispielen der Stadtverwaltungen München und Schwäbisch Hall, sowie eines mittelständischen Unternehmens, der Näve Leuchten GmbH, verschiedene Arten und Wege der Migration vorgestellt. In weiteren Kapiteln werden die politischen, sicherheitstechnischen, sowie monetären Dimensionen solcher Migrationen genauer analysiert.

2.2 Linux in München

Der Titel dieses Projektes ist LiMux. Es setzt sich aus den zwei Eigennamen München und Linux zusammen. Dieser Titel ist ausdrucksstark genug um intuitiv zu zeigen, um was es in diesem Projekt geht. Es soll die Verwaltung der Stadt München, die aus mehr als 14 000 Arbeitsplatzrechnern mit circa 16 000 Benutzerinnen und Benutzern besteht, auf ein Open Source Betriebssystem umgestellt werden. In der Verwaltung werden heute circa 300 Softwareprodukte mit 170 Fachverfahren, in 17 Organisationen mit eigenständiger EDV, eingesetzt. Dies führt zu unterschiedlichen Betriebskonzepten, Logistiken, Benutzerverwaltungen und Support. (nach Hoegner (2004)) Um diesem Durcheinander ein Ende zu setzen, ist ein Basissystem, in allen Bereichen gleich, die Lösung.

Um den Umstieg auf Linux zu verstehen, bedarf es einiger historischer Informationen. Der Standard-Arbeitsplatzrechner in der Verwaltung war mit Windows NT 4.0 als Betriebssystem, sowie der Microsoft Office-Suite ausgestattet. Da der Support von den eingesetzten Produkten Seitens des Herstellers eingestellt wurde, sowie neue Software bzw. Software-Versionen ausschließlich für die nachfolgenden Betriebssysteme erhältlich waren und die nun auf dem Markt kommende neue Hardware nur durch die neuen Betriebssysteme unterstützt wurde, herrschte ein starker Migrationszwang. (nach Hoegner (2004)) Die Umstellung musste entweder auf einen Windows NT 4.0 - Nachfolger oder eine andere Plattform erfolgen. Am 14.11.2001 verlangte der Stadtrat der Stadt München eine Vorlage zur Erfassung, ob es Alternativen zu Microsoft im Bereich des Betriebssystems sowie im Office-Bereich gibt. (Strobl (2004)) Am 17.11.2002 wurde eine Client-Studie in Auftrag gegeben, welche im Dezember 2002 abgeschlossen wurde. In ihr (Unilog Integrata (2003)) werden fünf Alternativen hinsichtlich Machbarkeit, Wirtschaftlichkeit sowie „strategisch-operativer Bewertung“ Unilog Integrata (2003) untersucht. Es wurden eine vollständige Microsoft Variante bestehend aus Windows XP und Office XP, Mischformen aus Windows XP mit OpenOffice¹, Linux und OpenOffice, als auch Varianten mit einem PC-Emulator sowie der Einsatz eines Terminalservers vorgeschlagen. Am 28. Mai 2003 wurde der Weg festgelegt, und man entschied für Open Source Software mit Webanwendungen. (Bielecke (2002)) Als Vorteil verspricht man sich unter anderem eine erhöhte Herstellerunabhängigkeit, dem erhöhtem Wettbewerb im Software-Markt als auch von der besseren Erreichbarkeit der strategischen Ziele der Stadt München. (nach Strobl (2004))

Zur weitestgehenden Nutzung der projektierten Vorteile, auch in der Zukunft, werden neue Client-Server Anwendungen ausschließlich als Webanwen-

¹eine freie Office-Suite

ung ausgeschrieben. Ferner werden Open-Source Betriebssysteme sowie Office-Suiten zum Einsatz kommen. Um das gesteckte Ziel der vollständigen Migration zu erreichen, gehen die Münchener ihren Weg schrittweise. In drei Phasen soll, bis zum Jahre 2008, alles angepaßt werden, die schwierigen Fachanwendungen hierbei zuletzt. (nach Bielecke (2002))

In diesem Jahr werden die 7000 Formulare und Makros migriert sowie sämtliche Computer mit Open Office und Mozilla als Browser ausgestattet. In den folgenden 2 Jahren werden die Client-Rechner auf Linux umgestellt und bekommen, bis zur vollständigen Migration, den Emulator Wine, um ebenjene Software weiterhin nutzen zu können.

Abschließend seien die Kosten in Höhe von circa 35 Millionen € erwähnt. (aus Unilog Integrata (2003)) Wobei 38 % der Gesamtkosten allein für Schulungskosten ausgegeben werden. Die Partner dieses großen Projektes sind IBM und Suse/Novell, zahlreiche kleine Firmen und die Universität München. (Bielecke (2002)) Die Kosten für eine reine Microsoft-Lösung waren laut Unilog Integrata (2003) zwar 4 Millionen € unter dem der Linux-Lösung. Die Münchener Stadträte entschieden sich aber dennoch für den Pinguin, da die beauftragte Unternehmensberatung Unilog den „qualitativ-strategisch“ höheren Mehrwert der Linux Lösung lobte. (nach Unilog Integrata (2003))

2.3 Linux in Schwäbisch Hall

Die Stadt Schwäbisch Hall liegt 70 km nördlich von Stuttgart und hat ca. 36 000 Einwohner. Die bis vor kurzem eingesetzte IT-Infrastruktur bestand hauptsächlich aus Windows NT 4.0 Arbeitsplatzrechnern. Die Ausgangslage verhält sich also ähnlich wie in der Stadt München. Jedoch nehmen hier Ausgaben für Lizenzen einen anderen Stellenwert im Haushalt ein, als in München. (Hilbert (2002)) In der IT sind 81 Software Produkte an insgesamt 17 Standorten zu finden. Es befinden sich 325 Rechner im Einsatz. Ein überschaubarer Bereich, in dem 4 Administratoren sowie 2 (Azubi-)Fachinformatiker die Migration durchführen. Der Anfang wurde 1997 mit der Umstellung der Server auf Linux mit den Serverdiensten wie z. B. samba, cups, mail und sqld vollendet. Es stellte sich heraus, dass die Lösung zu geringen Kosten führte, unter anderem durch die längere Nutzung der Hardware, geringe Softwarekosten und dem viel kleineren administrativen Aufwand. Nun war klar, dass man durch den Einsatz von Linux ein Mehrgewinn erfahren könnte. Die Clientseite wurde nun betrachtet, die Stadtverordneten wurden überzeugt und der Oberbürgermeister erklärte das Projekt „Linux in die Amtstube“ zu bringen zur Chefsache. (Hilbert (2002)) Die Bedingungen waren allerdings, daß keine Mehrkosten produzieren würden, die bisherige Fachsoftware weiterhin zum Einsatz kommen solle und die Kompatibilität zu vorhandene Lösungen gewährleistet bleiben müsse. (nach Kloiber (2003)) Die Umstellung verläuft erfolgreich. (siehe auch Hilbert (2003)) Zum jetzigen Zeitpunkt ist die Umstellung der Rechner auf 95 % vorangetrieben. (Braeuner (2004))

2.4 Linux in einem mittelständischem Unternehmen

Obwohl keine Behörde, sei hier abschließend noch das Beispiel der Migration der Serversysteme der Näve Leuchten GmbH vorgestellt. Dort verlief die Migration anders als zunächst geplant. Das Unternehmen hatte seine Server von Microsoft

Produkten auf OSS umgestellt. Nachdem die, durchaus erwarteten anfängliche Probleme der Migration gelöst waren, lief der Arbeitsbetrieb im normalen Rahmen weiter. Durch eine Entscheidung der Geschäftsleitung, die auf einer Messe ein Warenwirtschaftssystem erstand, kam es jedoch zu Problemen: Die Installation auf den Linux-Servern scheiterte. Um den Produktivbetrieb aufrecht zu erhalten, entschieden sich die zuständigen Administratoren für eine Remigration zu Windows. Diesmal kam ein Windows .NET 2003 auf dem Server zum Einsatz, zu teuer bezahlten Lizenzen. (Näve Leuchten GmbH (2004))

Eine Lehre, die man anhand dieser Projektvorstellung ziehen kann, daß der Erfolg der Durchführung einer solcher Migration vor allem von der vorhandenen Infrastruktur, genauen Vorstellungen seitens des Managements, der konkreten Realisierung sowie möglicher zukünftiger Optionen, abhängt.

3 Open-Source Software und Politik

3.1 Die Situation in Europa

3.1.1 England

Das britische Office of Government Commerce (OGC) hat einen Report Anfang November zum Thema OSS veröffentlicht. In dem Report wird OSS als „brauchbare und vertrauenswürdige Alternative mit dem Potential für Kosteneinsparungen“ bezeichnet. Jedoch möchte die britische Regierung nicht alle Systeme auf OSS umstellen, sondern die Möglichkeit in Betracht ziehen, es einzusetzen. Zudem wurden einige Projekte gestartet, unter anderem ein Projekt bei der schottischen Polizei. Als wichtige Partner sind Sun und IBM benannt worden.

3.1.2 Frankreich

Bernard Benhamou, zuständig für Zukunftsfragen und Internetregulierung im Bereich der E-Government-Entwicklung im Büro des französischen Premierministers, hielt ein Vortrag auf der Konferenz „Computers, Freedom & Privacy“. Dort wurde die Meinung, das OSS ein sehr wichtiger Teil einer demokratischen Gesellschaft ist, stark verteidigt.

„Für die meisten Systeme brauchen wir dafür Gewissheit, dass die ausgetauschten Informationen Bestand haben und unter demokratischer Kontrolle bleiben. [...] Der Zugang zu den Daten muss auch in 100 Jahren noch möglich sein. Proprietäre Formate sind dabei hinderlich.“ (Bernard Benhamou)

Man wolle die Lizenzkosten für die proprietäre Software einsparen und somit die Möglichkeit haben, die OSS Systeme in mehreren Sprachen übersetzen zu können.

3.1.3 Spanien

Obwohl es keine größeren Meldungen über OSS Einsatz in den Medien gibt, laufen derzeit mehrere Projekte, die von der zentralen und regionalen Regierungen gesponsort werden. Der grösste davon ist „Virtual MAP“ der von dem Ministerium der öffentlichen Administration ins Leben gerufen wurde. Die Implementation basiert auf 400 Servern mit mehr als 4000 simultan angeschlossenen Klienten (Ghosh u. a., 2002). Dazu werden Schulen in den ärmeren Regionen des Landes mit OSS Netze ausgerüstet, um die teureren Lizenzen von proprietären Software einzusparen.

3.2 Die Situation in Deutschland

3.2.1 Die Bundesregierung

Die Bundesregierung will sich bei dem Fördern von Klein-/Mittelunternehmen stark machen. In ihrem Kapitel VIII „Sicherheit, Toleranz und Demokratie“ des Koalitionsvertrages zwischen SPD und Grüne, steht folgendes:

„Das Urheberrecht werden wir bezogen auf neue Technologien fortentwickeln. Die Rechtsordnung muss die Wettbewerbsstellung kleiner Betriebe im Software-Bereich stärken. Open-Source-Produkte dürfen nicht benachteiligt werden.“ (Koalitionsvertrag SPD/Grüne, Kapitel VIII)

Beim Parteitag der SPD in Bochum (2003) wird das Thema OSS auch kurz besprochen.

„Aus IT-Sicherheitsperspektive wie auch aus wirtschaftspolitischer Perspektive kommt zudem der Förderung von Open-Source-Software eine weitaus größere Bedeutung zu.“

Die SPD sieht in Open-Source Software eine einzigartige Chance für die Wirtschaft Europas:

„Open-Source stellt eine besondere Chance für die europäische Softwarebranche dar. Zum ersten Mal gibt es hier ein Feld, in dem die USA nicht führend ist. Diese Chance muss genutzt werden. Das Bundesministerium für Wirtschaft und Technologie hat bereits 1999 die Fortentwicklung von Open-Source Sicherheitskomponenten gefördert.“

Somit sieht die Bundesregierung eine politisch interessante und wirtschaftlich wichtige Alternative zu proprietärer Software.

3.2.2 Migrationsleitfaden

KBSt, Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, hat im Juli 2003 ein 440 Seiten langen Migrationleitfaden zusammengestellt (KBSt, 2003). Darin wird die Meinung verteidigt, dass OSS „eine reife Alternative zu proprietärem Software darstellt. Die steigende Tendenz OSS in der Unternehmensebenen einzusetzen, spricht über die wirtschaftlichen Vorteile deutlich genug. Jedoch werden die Möglichkeiten OSS zu realisieren, von erheblichen wirtschaftlichen Interessen stark beeinflusst.“

3.2.3 Justitia online

Die Bundesregierung hat die „Verordnung über den elektronischen Rechtsverkehr beim Bundesverwaltungsgericht und beim Bundesfinanzhof“ erlassen. Damit kann ab dem 1. Dezember 2004 der Schriftverkehr mit diesen beiden Gerichten rechtswirksam auch in elektronischer Form abgewickelt werden. Hier wird auch auf Open-Source Software gesetzt und zwar Dokumente die mit „Open Office“ erstellt worden sind, sind ausdrücklich zugelassen. Zusätzlich müssen alle Dokumente mit einer digitalen Signatur nach dem Signaturgesetz signiert werden.

3.2.4 IBM/Microsoft Rahmenverträge

Die Bundesregierung hat mit IBM einen Rahmenvertrag für den Einsatz von Linux im öffentlichen Sektor unterzeichnet. Die Gründe dafür waren unterschiedlich, wie z.B. der gesunde Wachstum von Linux im Server-Bereich, die Sicherheit durch offene Quellen und die Anfälligkeit von Monopolstrukturen wie Microsoft.

Zusätzlich wurde betont, dass die Abwesenheit von einer Alternative im Bereich Preisbildung, ein seriöser Nachteil ist, da die Preisbildungspolitik von den Monopolisten doch hindernd ist. Trotzdem wurde auch ein Rahmenvertrag mit Microsoft unterzeichnet, der eine starke Änderung im Bereich Lizenzkosten im Vordergrund stellt.

3.2.5 CDU

Eine wichtige Voraussetzung für die Einsetzung von Open-Source Software in der öffentlichen Verwaltung ist die Transparenz des Quellcodes. Dies fördert die IT-Sicherheit. Man muss herstellerneutral gegenüber den Mitbürgern bleiben.

„Zur Förderung von Open-Source Software sollten öffentliche Stellen nur Software verwenden, deren Quellcode frei zugänglich ist, soweit solche Software verfügbar ist.“ (Chancen@Deutschland, Eine Internetstrategie für die Politik, S.3)

3.2.6 FDP

„Linux - soweit es vernünftig ist.“

Die FDP spricht positiv über die Benutzung von Open-Source Software in der Verwaltung. Die Partei setzt Linux als E-Mail und Groupware-Server selbst ein. Trotzdem gibt es auch Punkte wo Kritik notwendig ist:

„Was wir kritisieren, sind Linux Anwendungen, die sich über die objektiv nachprüfbar Kriterien Bedienerfreundlichkeit, Stabilität und Kosten hinwegsetzen, um politisch erwünschte Signalwirkungen zu erreichen. Das Betriebssystem des Bundestages ist auf optimale Funktionsfähigkeit angewiesen und eignet sich nicht für ideologische Schaukämpfe.“ (Hans-Joachim Otto, FDP)

3.2.7 PDS

Der wichtigste Standpunkt von der PDS ist deren Position gegen Softwarepatente. Ein offener Quellcode ist die Voraussetzung für Weiterentwicklung und IT Sicherheit:

„Die PDS setzt sich für Open-Source Software und gegen Softwarepatente ein. Die Offenlegung der Quellcodes ermöglicht eine Vergesellschaftung der Programme und damit ihre stetige Verbesserung und Verbreitung. [...] Auch Sicherheitsgründe sprechen für Open-Source Software. Die Funktion und Qualität einer Verschlüsselungssoftware etwa ist nur bei Offenlegung der Quellcodes möglich.“

3.2.8 Treffen in Berlin, 10.11.2004

Beim Treffen in Berlin am 10.11.2004 haben sich mehrere Vertreter der Politik und Wirtschaft zusammengefunden um über rechtliche, ökonomische und wettbewerbspolitische Aspekte von proprietärer Software und Open-Source Software zu diskutieren. Zwar waren die Teilnehmer einig, dass OSS den Wettbewerb intensivieren kann, aber es gab auch andere Meinungen, wie die von Stefan Kooths

vom Münster Institute for Computation Economics (MICE) der Universität Münster:

„Open-Source Software schafft keine Wertschöpfungspotenziale, sondern bietet nur einen Teil der Möglichkeiten des kommerziellen Marktes. Die Umsatz- und Beschäftigungseffekte von Open-Source Software sind damit geringer als die Effekte kommerzieller Software.“

Auch juristisch ist die Situation von OSS noch nicht klar genug. Eine Überlassung gilt als Schenkung, trotzdem gibt es bei den Definitionen Schwierigkeiten, meinte Gerald Spindler, Professor an der Universität Göttingen:

„Denn trotz Schenkung bleiben die Hersteller zumindest in bestimmten Fällen Dritten gegenüber haftbar. Und auch der Käufer selbst hat gegenüber Händlern und Vertrieben einen Haftungs- und Gewährleistungsanspruch.“

Das Thema Patentierbarkeit computerimplementierter Erfindungen wurde im Verlauf der Diskussion auch angesprochen. Fast alle Vertreter beider Seiten, sowohl Politik als auch Wirtschaft, waren sich einig, dass derartige Patente auch in der Zukunft erteilt werden sollen. Trotzdem müsste man sicherstellen, dass man keine Trivialpatente oder „Patente auf reine Geschäftsmethoden“ erteilt. Dazu ist noch wichtig zu sagen, dass die Förderung von OSS nicht zulasten von proprietärer Software stattfinden soll. Die Vertreter der Politik waren der Meinung, dass man sich zu keiner der beiden Seiten binden möchte, aber alle Möglichkeiten offen halten möchte. In dem Zusammenhang ist eine Änderung (Verschärfung) des Wettbewerbsrecht nicht zwingend notwendig.

3.3 Fazit

Die Bundesregierung bemüht sich die Open-Source Branche zu stimulieren. Jedoch ist es schwer sich für die eine oder die andere Seite fest zu entscheiden. Hier tritt die Bundesregierung gleichzeitig als Regulator und Klient auf. Interessant sind die Aussagen, in welchen betont wird, dass man die kleinen und mittleren Unternehmen stimulieren möchte, aber trotzdem Rahmenverträge mit großen Unternehmen wie Sun und IBM abgeschlossen werden. Der Rest der politischen Szene fährt einen ähnlichen Kurs. Man möchte unbedingt die Freiheit haben, den Quelltext des Software Produktes sehen zu können, aber trotzdem kann man sich nicht zwischen Open-Source und proprietärem Software entscheiden. Man lässt sich die Möglichkeit zwischen den beiden Konzepten frei zu wählen und somit pflegt man gute Beziehungen zu beiden „Fronten“, deutlich bei den vielen Treffen von Wirtschafts- und Politikvertretern zu sehen. Allgemein zu sagen ist es, dass die Situation momentan klar unklar ist. Mit der dauernden Diskussion über Softwarepatenten, wo die politischen Kräfte eine sehr ernste Rolle spielen, ist abzuwarten wie das Dilemma gelöst wird. Es gibt viele Beispiele für den Einsatz von Open-Source Software im öffentlichen Sektor, aber es gibt auch sehr viele Beispiele wo keine Migration geplant ist und ein Wechsel fast unmöglich aussieht, manchmal auch gar nicht gewünscht ist.

4 Der Kostenaspekt

4.1 Migration

4.1.1 Migrationsgründe

Migrationen, sprich der Wechsel von einem Systems auf ein anderes, sind natürliche Ereignisse im „Leben“ eines IT-Systems - sei es durch das Auslaufen von Lizenzen, den Wunsch bzw. die Notwendigkeit des Ausbaus eines Systems, die Verfügbarkeit neuer Hard- und Software nur für neue Betriebssystemversionen, die Unterstützung neuer Schnittstellen oder durch „strategische Ziele, wie beispielsweise verstärkte Herstellerunabhängigkeiten und erhöhte Interoperabilität“ (KBSt (2003), S.19) bedingt. Mit einer solchen Migration sind jedoch Kosten verbunden, die sich je nach gewähltem Migrationsweg zum Teil erheblich voneinander unterscheiden können. Eine fundierte Kostenabschätzung ist daher vonnöten.

4.1.2 Migrationswege

Nach KBSt (2003), S.363ff gibt es im wesentlichen 3 mögliche Szenarien für ein derartiges Vorhaben

- Ablösende Migration
- Fortführende Migration
- Teilmigration (punktuelle Migration)

Vollständige Migration

Eine im Sinne obiger Quelle „vollständige Migration“ bezeichnet die *vollständige* Umstellung *aller* Systeme, wie z.B. Datenablage, Druckdienste, Authentisierungsdienste, Netzwerkdienste, System-Überwachungs- und Management-Dienste, Verzeichnisdienste, die Middleware, Web Services, Webserver, Portallösungen, Dokumentenmanagementlösungen, Datenbanken, Groupware, Terminal-Server und Thin Clients, Hochverfügbarkeitslösungen sowie nicht zuletzt der Bereich Office/Desktop auf ein anderes Betriebssystem.

Fortführende Migration

Unter „fortführender Migration“ hingegen versteht man die Ablösung eines (Software-)Systems durch ein Nachfolgeprodukt.

Teilmigration (Punktuelle Migration)

Die letztgenannte Variante ist eine Mischform, bei der für einzelne Teilsysteme eine der beiden erstgenannten Formen durchgeführt wird (z.B. Wechsel der Serverbetriebssysteme).

4.1.3 Migrationsarten

Derartige Migrationsvorhaben (vollständig wie fortführend) lassen auf zwei Arten bewerkstelligen² (KBSt (2003) S.383ff): *Schnell* bzw. *sanft*. Beide Arten mit spezifischen Vor- und Nachteilen, sowie unterschiedlichen Anforderungen an die Beteiligten.

Schnelle Migration

Bei der „schnellen Migration“, die innerhalb eines *festen* Zeitrahmens erfolgen soll, wird versucht unter Beibehaltung eines funktionalen Systemzustandes, in nur *einer*³ Phase den Umzug von einem System auf das neue zu vollziehen. Dies ist eine sehr anspruchsvolle Aufgabe, die hohe Anforderungen bezüglich der Planung des Projektes, der beanspruchten Finanz- und Zeitmittel, des technischen Personals, die Qualifizierung desselben und nicht zuletzt an die Nutzer stellt, die sich auf das neue System einzustellen haben.

Als Gründe für eine schnelle Migration sind unter anderem zu nennen: Pure Notwendigkeit, bedingt durch ein Auslaufen von Lizenzen, der Wunsch nach einem nur kurzzeitigen Betrieb eines heterogenen System, sowie die nur einmalige Konfrontation mit einem neuen System (sowohl für Administratoren, als auch Benutzer). Ein derartiges Vorgehen ist dann möglich, wenn im vorliegenden System nicht zu viele Dienste miteinander „verwoben“ sind und adäquater Ersatz für Fachanwendungen vorliegt - nicht zuletzt sind bereits vorhandene Kenntnisse der Administratoren das Zielsystem betreffend nicht von Nachteil.

Sanfte Migration

Im Gegensatz zur schnellen Migration gibt es bei der „sanften Migration“ keinen festen Zeitplan, sondern nur einen grob definierten Zeitrahmen, in dem die Migration stattfinden soll. „Sanft“ bezieht sich hierbei darauf, daß stückweise Komponenten des alten Systems gegen neue ausgetauscht werden. Idealerweise erfährt der Nutzer des Systems keinerlei Änderung des Systemverhaltens. Im Allgemeinen wird man zuerst die Server-Infrastruktur ersetzen, gefolgt von einer eventuellen Officemigration⁴ Typische Vertreter eines ersten Umtauschs wäre z.B. der e-Mail-Server, dessen Verhalten unabhängig vom Client ist.

Gründe für diese Migrationsart sind u.a. eine mögliche Anpassung der anfallenden Kosten an die jeweilige Haushaltslage, sowie die Möglichkeit komplexe Systeme schrittweise aufzulösen. Eventuell fehlendes Know-How kann auf diese Weise Schritt für Schritt aufgebaut werden und spätere Projektphasen dadurch beschleunigen. Der Ablauf und die Phasen einer solchen Migration wird hier beispielhaft an den Abbildungen 1 und 2 auf der nächsten Seite dargestellt (jeweils entnommen aus: KBSt (2003), S.386f).

Solche Migrationen können unter Umständen sehr groß ausfallen: Allein in der

²für den Fall, daß eine punktuelle Migration stattfindet, gilt für das jeweilige System analoges

³Der Terminus „schnell“ mag daher etwas irreführend erscheinen.

⁴KBSt (2003) S.387 nennt hier als Beispiel den Einsatz von OpenOffice.org bzw. StarOffice auf der Windows-Plattform

bayrischen Verwaltung waren Ende 1999 mehr als 70000 PCs im Einsatz, die nach Planung auf über 110000 anwachsen sollen.(vgl. Bayrischer Oberster Rechnungshof (2001), S.63) Bei derartigen Vorhaben spielen nicht zuletzt Kostenaspekte eine Rolle.

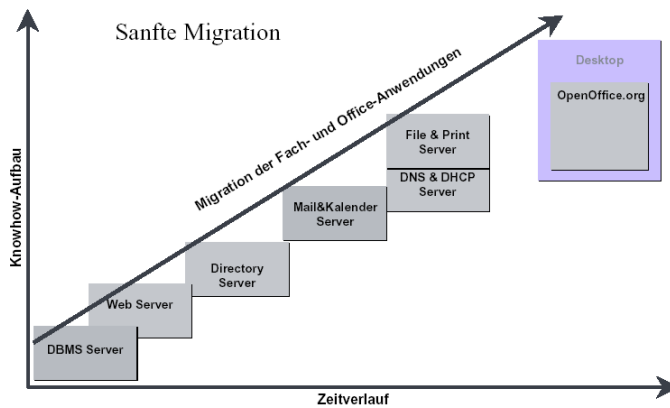


Abbildung 1: Möglicher Ablauf einer sanften Migration zu OSS mit OpenOffice (nach KBSt (2003), S.386f)

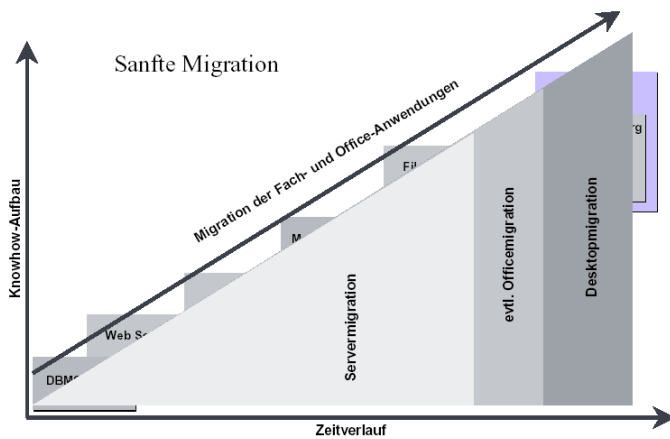


Abbildung 2: Phasen eines solchen Ablaufs (nach KBSt (2003), S.386f)

4.2 Kosten

Migrationsprojekte, wie die bereits vorgestellte Migration der Stadt München (siehe auch München (2004)) auf OSS, bewegen sich in einem nicht zu unterschätzenden finanziellen Rahmen. Eine Studie der Unilog Integrata (2003), S.17f im Auftrag der Stadt München veranschlagt die Kosten der Umstellung der Clients je nach gewählten Szenario im Bereich von 34,2 Mio.€ (fortführende Migration auf Office XP und Windows XP) bis 50 Mio.€ (Benutzung von Linux/OSS und Terminalservern) bewegen - bei Kapitalwerten von 31,3 Mio.€ bis 46,6 Mio.€.

Dieses Beispiel allein zeigt, daß solcherlei Entscheidungen wohlbedacht getroffen werden sollten. Zum Vergleichen der Kosten der Modelle kann man sich verschiedener alternativer Techniken bedienen: Die monetäre Analyse, die Nutzwertanalyse, die IT-WiBe 21, die sogenannte „Total cost of ownership“-Methode, sowie weitere.⁵

4.2.1 Monetäre Analyse

„Für die monetären Auswirkungen der Vorhaben wird die Kapitalwertmethode angewandt. Als dynamisches Verfahren beurteilt sie Investitionsprojekte nach ihrem Kapitalwert, d.h. durch wirklichkeitsnahe Erfassung der mit der Investition zusammenhängenden Finanzströme [...], fokussiert auf einen gemeinsamen Bezugszeitraum. Einnahmen und Ausgaben, die mit dem Vorhaben zusammenhängen können für fünf Jahre im Voraus geplant werden. Für die künftigen Werte wird der aktuelle Zeitwert durch Abzinsung [...] ermittelt.“ (KBSt (2003), S.295)

4.2.2 Nutzwertanalyse

„Gilt es dagegen bei der Entscheidungsfindung nicht monetär erfassbare Auswirkungen mit einzubeziehen, wird die Nutzwertanalyse angewandt. Sie bewertet einzeln und unabhängig voneinander gewichtete Zeilkriterien, um sie anschließend zu einer Gesamtbewertung zusammenzufassen. Hier werden die sogenannten „weichen“ Faktoren über Bewertungsskalen quantifiziert. [...]“ (KBSt (2003), S.295)

4.2.3 IT-WiBe 21

„In der Bundesverwaltung richten sich Wirtschaftlichkeitsbetrachtungen nach den Vorschriften des § 7 der BHO und nach den hierzu erlassenen Verwaltungsvorschriften, die seit 1995 vor allem betriebswirtschaftliche Verfahren berücksichtigen. [...] Die IT-WiBe ist ein Verfahren, mit dem anders als bei der TCO-Methodik nicht nur der Kostengesichtspunkt betrachtet wird, sondern in die Berechnung auch mögliche Ersparnisse einbezogen sind.“ (KBSt (2003), S.295f)

⁵Hier wäre z.B. das „return on investment“ zu nennen, das zusätzlich zur Aspekte der Leistungsfähigkeit eines Investitionsguts miteinbezieht (Z.B. Anzahl der gleichzeitig bedienbaren Clients eines Servers).

4.2.4 Total cost of ownership

Eine weitere Möglichkeit die Kosten einer Entscheidung für oder gegen ein bestimmtes Migrationsmodell abzuschätzen bietet die sogenannte *Total cost of ownership* kurz *TCO*. Das TCO-Modell ist umstritten: Im Jahr 1987 vom amerikanischen Analysten Gartner Group entwickelt, die feststellt, daß die

„[...] Kosten einer IT-Infrastruktur [...] und die im laufenden Betrieb entstehenden Kosten zu intransparent sind[,]“ (Wild und Herges (2000), S.5)

hat es sich inzwischen nahezu zur Standardherangehensweise zur Kostenanalyse der IT entwickelt. Die Wichtigkeit dieses Modells beweisen nicht zuletzt die zahlreichen Studien zur TCO im Auftrag großer IT-Firmen, wie Sun, IBM, Redhat oder Microsoft.

Nach einer Websuche bei Sun findet man über 1500 Treffer zu „TCO“, darunter ein eigene Seiten zur Berechnung der TCO spezifischer Produkte (Z.B. SUN (2004)). Weiterhin verlinkt allein Microsoft auf einer eigens dafür eingerichteten Seite (MS (2004)) zu mehr als 25 Studien (Stand: Nov.2004), die sich mit diesem Thema auseinandersetzen.

Ein häufiger Kritikpunkt ist die mangelnde Vergleichbarkeit zweier Studien auf Grund zu unterschiedlicher Voraussetzungen. Wild und Herges (2000), S.7f führt an:

„Diese Unterschiede weisen, von Verzerrungen bei den Schätzungen abgesehen, darauf hin, daß den Modellen divergierende Anforderungen und Bezugsgrößen für das jeweilige IT-Investitionsgut zugrunde liegen müssen. Ein Sachverhalt, der bei der Interpretation solcher Zahlen explizit berücksichtigt werden muß.“

Nach Wieland (2004), S.126 bezeichnet man nun mit der TCO die „[...]Summe der gesamten Kosten eines Investitionsgutes[...]“. Diese kann man in direkte und indirekte Kosten unterteilen.

Diese Kosten teilen sich auf (vgl. Wild und Herges (2000), S.15) in Kosten für:

- Soft- und Hardware
- Technischer Support
- Planungs- und Prozeßmanagement
- Verwaltungs- und Finanzaufgaben
- Schulung der Mitarbeiter (EDV-Personal und Endanwender)

Die Maßgröße hierbei sind Abschreibungen und Leasinggebühren im Bereich der Soft- und Hardware, sowie Löhne, Gehälter und Gebühren für Dienstleistungen in den anderen Bereichen.

Hierbei sind die Soft- und Hardwarekosten am leichtesten, nämlich als Summe der Kaufpreise, eventueller Upgrades und Lizenzkosten, zu berechnen sind. Kosten im Bereich des Supports (Installation, Wartung und Fehlerbehebung von in- wie externer Seite, sowie ggf. Literatur) und Personalkosten (Management, Systementwicklung, Administration, Training) genau anzugeben, fällt dagegen schwerer. Gerade dieser Punkt ist jedoch sehr wichtig: KBSt (2003), S.308 führt aus, daß die Personalkosten bei einer vollständigen Migration bis zu 90% der Gesamtkosten ausmachen, mit leichten Vorteilen für große Behörden - hier liegen die Kosten bei „nur“ 87% der Gesamtkosten.

Die indirekten Kosten bestehen im wesentlichen aus den Kosten, die durch (ungewollte) Ausfälle des Systems entstehen, sowie den Kosten des Produktivitätsverlustes, die durch die Einführung des Systems in der Belegschaft entstehen. Wild und Herges (2000), S.15 etwa nennt folgende Punkte:

- Schulungsmaßnahmen (formales Lernen)
- Lernen im Arbeitsalltag
- Self-Support sowie Peer-to-Peer-Support
- Datenverwaltung
- Entwicklung von Software
- (Futzing)⁶
- Downtime (geplant/ungeplant)

Die Maßgröße ist hier der Produktivitätsverlust (entgangene Löhne und Gehälter oder entgangener Umsatz).

Diese „versteckten“ Kosten können im Allgemeinen schwer beziffert werden. Kenwood (2001) drückt dies so aus:

„Indirect costs include “hidden” influences and causal inter-relationships that may be difficult to capture. Since they are potentially significant, they are important to identify and consider. These indirect costs can be measured in terms of lost productivity attributed to the computing environment. While the salary and other labor costs associated with an employee are captured under the direct cost category, the indirect costs represent labor costs that are “wasted” and could be used in more productive ways. In other words, although there is no additional direct cost to the organization, not as much output was received from the employee due to inefficiencies in the process or system.“

⁶Die Gartner Group zählt zu den indirekten Kosten ebenfalls den sogenannten „Futz-Faktor“ - ein Maß für die Nutzung der IT zu privaten Interessen, also z.B. Surfen oder Chatten (siehe Wild und Herges (2000), S.14) - dieser Punkt ist jedoch umstritten und taucht in anderen Modellen zur TCO nicht auf

Ein weiterer möglicher Kostenpunkt, der hier nicht aufgeführt wurde, ist das eventuelle Risiko einer Patentverletzung durch den Einsatz von OSS. Der SCO-Fall⁷ sorgte hier nicht zuletzt für einige Unsicherheit. Demgegenüber führt ein (Kurz-)Gutachten der Kanzlei Frohwitter (Sedlmayer und Gigerich (2004), S.38) im Auftrag der Stadt München aus, daß:

„Bislang [...] kein Fall bekannt [ist], in dem ein Patent gegen Linux eingesetzt worden wäre. Dies, obschon Linux seit über 10 Jahren benutzt wird.“

Sedlmayer und Gigerich (2004) fährt fort:

„Wie [...] dargelegt, ist das Risiko der Patentverletzung bei proprietärer Software und OS-Software im Wesentlichen gleich.“

Da inzwischen sogar Versicherungen gegen derartige Unwägbarkeiten existieren⁸, kommt diesem Aspekt daher eine eher untergeordnete Wichtigkeit zu.

Doch welcher Faktor stellt nun den größten „Kostentreiber“ dar? Unilog Integrata (2003) kommt zu dem Schluß, daß unabhängig vom gewählten Migrationsweg die Bereiche Schulungsteilnahme und Einarbeitung, sowie die Kosten für Trainer, Räume und Infrastruktur den größten Teil der Kosten ausmachen: bis weit über 50% (siehe dazu auch die Abbildungen 3, 4).⁹

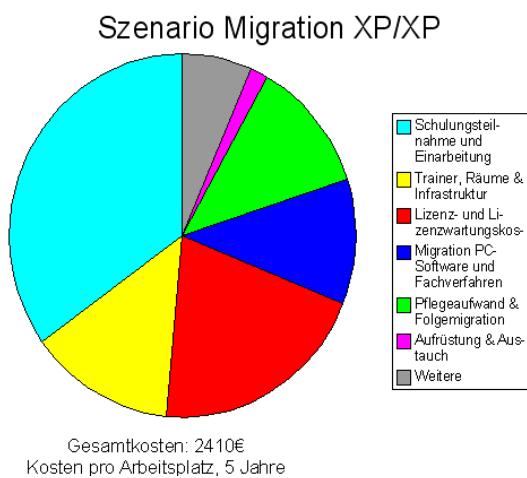


Abbildung 3: relative Kosten für den Einsatz von Office XP/Windows XP

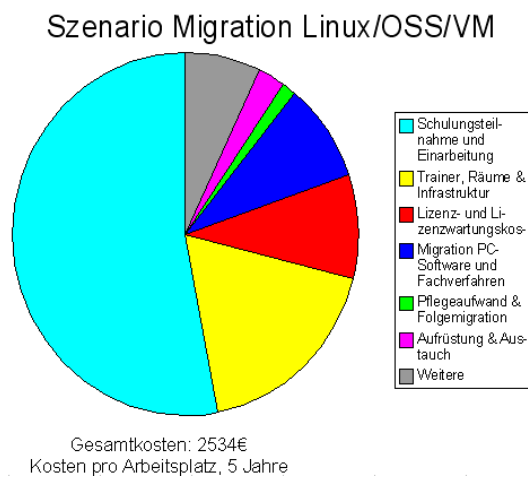


Abbildung 4: relative Kosten für den Einsatz von Linux/OSS/VMware

⁷ zur Historie dieses Falles siehe z.B. Jones (2004)

⁸ z.B. vom New Yorker Start-Up „Open Source Risk Management“ siehe: Open Source Risk Management (2004)

⁹ jeweils erstellt nach den Daten aus Unilog Integrata (2003), S.37f - VMware ist ein sog. Emulator mit dem (Windows-)Anwendungen auch unter Linux benutzt werden können

4.3 Conclusio

Welcher Weg ist nun also für wen der beste? Hier herrscht eine geradezu babylonische Verwirrung. Varian und Shapiro (2003), S.13f führen aus:

„There have been several attempts to compare the TCO of Windows and of Linux in various computing environments. In most of the studies the difference in TCO is on the order of 10 or 15 percent. This difference is not large; a 10 percent difference in TCO could easily be swamped by local conditions, random events, and other considerations. To a first approximation, it seems reasonable to suppose that neither of these two platforms has a striking advantage over the other in terms of conventional measures of TCO.“

Weiterhin:

„Users should be very wary of adopting a system that will be difficult to switch *away from* in the future, in part because the lock-in associated with using such a system will reduce their future bargaining power with their vendor.“

Abschließend bleibt festzustellen, daß je nach Szenario beide¹⁰ Seiten (Windows wie Linux/OSS) ihre spezifische Vorteile ausspielen können. Daher erscheint bei allen Entscheidungen, die die IT-Kosten betreffen, eine fundierte Kostenabschätzung geboten.

Oder wie Barr (2002) sich ausdrückt:

„For one thing, TCO is like fine wine: it doesn't travel well. What may be true in one situation is reversed in another. What gets trumpeted as a universal truth [...] may or may not be true in a specific case, but it is most certainly false when claimed universally...“

¹⁰wobei natürlich auch weitere Alternativen, wie z.B. dedizierte UNIX-Systeme, die hier nicht betrachtet wurden, Beachtung verdienen

5 Sicherheit von Open-Source-Software

In der von der Stadt München beauftragten Client-Studie wurde neben Aspekten wie der Wirtschaftlichkeit auch hoher Wert auf die Sicherheit des Softwaresystems gelegt (Unilog Integrata, 2003).

Da die Sicherheit von Software allgemein und grade auch in der Verwaltung von hoher Wichtigkeit ist, wird auf diese mit Konzentration auf den Open-Source-Bereich in den folgenden Abschnitten näher eingegangen werden.

5.1 Grundlegende Merkmale kommerzieller und freier Softwareentwicklung

Mehrere Faktoren wirken auf die Entwicklung kommerzieller Software ein; so besitzt ein Projekt einen Zeitrahmen, der eingehalten werden muss, ein bestimmtes Budget und Ziele, wie die Profitmaximierung und das Ausschalten der Konkurrenz. Diese Interessen werden oft zum Nachteil der Softwarequalität und damit auch der Sicherheit des Produktes durchgesetzt (Gehring, 2001).

Um einen großen Gewinn zu erwirtschaften, kann die Fehlerfreiheit von Software ein gewisser Wettbewerbsvorteil sein, jedoch ist die Ausprägung der Realisierung dieses Aspekts von ökonomischen Interessen abhängig. Erschwerend kommt hinzu, dass die Sicherheit von Software nicht richtig belegt werden kann. Um ein Produkt konkurrenzfähig zu halten, fällt die Entscheidung meist auf die Erhöhung der Programmfunktionen, die sich leicht in der Produktbeschreibung anführen lassen, weniger auf die Beseitigung von Fehlern.

Die an Open-Source-Projekten beteiligten Personen sind unabhängig von ökonomischen Interessen und vom Marktdruck, sie können sich daher hauptsächlich auf die technischen Aspekte der Softwareentwicklung konzentrieren. Die Hauptmotivation der Mitglieder von Open-Source-Projekten ist die Vielzahl von persönlichen und sozialen Vorteilen (Gehring u. a., 2004, S.346), die durch die Zusammenarbeit in einer Gemeinschaft von Gleichgesinnten entsteht. Der Arbeitszwang bei der bezahlten Tätigkeit in einem Unternehmen entfällt, da die meisten Mitglieder freiwillig in ihrer Freizeit für das Projekt tätig sind.

„There’s one lesson that’s really obvious: You cannot motivate the best people with money. Money is just the way to keep score. The best people in any field are motivated by passion. This becomes more true the higher the skill level gets.” (Raymond, 1997)

5.2 Open-Source-Entwicklungsmethoden wirken sich auf die Codequalität aus

Das Grunddesign von Open-Source-Projekten wird meist einzeln oder in kleinen Gruppen erarbeitet. Handelt es sich nicht um ein von Grund auf neues Programm, so kann auf den Code von anderen Open-Source-Projekten zurückgegriffen werden. Hier kommt ein wichtiger Punkt der Open-Source-Philosophie zum Ausdruck: Die Wiederverwendung von bereits bewährten und oft getesteten Code als Fundament für die eigene Softwareentwicklung. Dieses Vorgehen erhöht die Sicherheit und die Entwicklungsgeschwindigkeit.

Erst wenn das Werk einen gewissen Reifegrad erreicht hat, wird es der Internetgemeinschaft über Newsgroups angekündigt und zum Download bereitge-

stellt. Neben der Attraktivität des Projekts für andere Entwickler spielt auch die Struktur des Projekts eine Rolle, ob sich viele Programmierer an der Entwicklung beteiligen werden. Stark modularisierte Projekte haben dabei bessere Chancen als monolithisch organisierte (Gehring u. a., 2004, S.181).

Bei Closed-Source-Projekten können Nutzer und Entwickler meist nicht direkt in Kontakt treten. Fehlermeldungen von Seiten der Softwarenutzer erfolgen über den Support, der die Meldungen über weitere Instanzen in der Firmenhierarchie weiterleitet, wobei Informationsverlust auftreten kann.

In Open-Source-Projekten ist die ausgeprägte Kommunikation ein elementarer Bestandteil und eine Notwendigkeit für den Erfolg einer Entwicklung (Gehring u. a., 2004, S.187). Die Kommunikation zwischen den beteiligten Personen wird durch Webseiten, Mailing-Listen, Newsgroups und Entwicklertools wie das CVS¹¹ ermöglicht. Bei größeren Projekten kommen noch Fehlerdatenbanken hinzu. Die Verwendung des CVS hat zusätzlich den Vorteil, dass jede Codezeile einem Autor zugeordnet werden kann, so dass Personen mit schlechten Absichten ermittelt werden können. Ein weiterer Kommunikationsweg ist die Codekommentierung. Die Entwicklung in einem Open-Source-Projekt hält die Beteiligten zur disziplinierten Codekommentierung an; diese ermöglicht anderen Programmierern nicht nur den schnellen Einstieg in die Codefunktionalität, sondern erleichtert auch die Suche nach fehlerhaften Zeilen im Code.

Die Stabilität freier Software resultiert bei größeren Projekten aus der hohen Anzahl von Anwendern und Entwicklern, die eine umfangreiche Fehlerlokalisierung und -behebung ermöglichen. Ein etabliertes Verfahren zur Fehlersichtung ist das aus der Wissenschaft bekannte *Peer Review* (Gehring u. a., 2004, S.212). Es beschreibt die Prüfung des Codes durch Programmierer auf der ganzen Welt.

Um den Werdegang der Prüfung nicht ganz dem Zufall zu überlassen, kann er durch einige befugte Personen beaufsichtigt werden. Jede dieser Personen hat die Verantwortung über einen bestimmten Codeabschnitt. Gefundene Fehler und erstellte Patches werden an die Aufseher weitergeleitet und nach erfolgter Prüfung in das CVS gestellt oder bei größeren Projekten der nächsthöheren Instanz von Kontrolleuren übergeben. Dieses Vorgehen wurde bei der Entwicklung des Linux-Kernels praktiziert.

Die Leistungsfähigkeit des Peer Reviews zeigt sich bei größeren Open-Source-Projekten: Hintertüren werden meist schnell gefunden und beseitigt. Ein bekanntes Beispiel ist Borlands Datenbankpaket Interbase, welches über einen Zeitraum von sechs Jahren eine von den Entwicklern integrierte Hintertür enthielt (Kamat, 2004). Diese entdeckte man kurz nachdem der Code des ursprünglichen Closed-Source-Projekts freigegeben wurde.

„Given enough eyeballs, all bugs are shallow.” (Raymond, 1997)

Es müssen nicht unbedingt extrem talentierte Programmierer an den Projekten beteiligt sein:

„The quality of open-source-software programmers does not need

¹¹Concurrent Versions System (CVS) bezeichnet ein Programm zur Versionsverwaltung von Quellcode. CVS vereinfacht die Verwaltung von Quellcode insofern, dass es alle Dateien eines Software-Projektes an einer zentralen Stelle speichert. Somit behält man den Überblick über die einzelnen Versionen der Dateien und die dazugehörigen Kommentare. Es ist zudem in der Lage, besonders bei größeren Projekten, die Arbeit der einzelnen Entwickler eines Projektes zu koordinieren.

to be as high as closed-source-software ones in order to achieve a bug-free state rapidly.”(Challet u. a., 2004)

Stetiges Verbessern, Ergänzen und Fehlerbeseitigen verlangt nach ebenso stetigen Veröffentlichungen. Ein Motto der Open-Source-Bewegung ist deshalb: „Release early, release often” (Gehring u. a., 2004, S.183). Dadurch ist gewährleistet, dass sich Fehlermeldungen auf den Code beziehen, an dem die Entwickler aktuell arbeiten und nicht schon umfangreiche Veränderungen seit der zuletzt veröffentlichten Version vorgenommen wurden, die ein Beseitigen der Fehler erschweren (Challet u. a., 2004).

5.3 Umgang mit Sicherheitslücken

Ein Nachteil von geschlossenen Code ist, dass der Anwender vollständig vom Hersteller abhängig ist. Beim Auftreten einer Sicherheitslücke muss der Nutzer warten bis der Softwarehersteller ein Patch bereitstellt, bis dahin ist das System des Anwenders für Angriffe verletzlich. Ob die Sicherheitsmängel überhaupt beseitigt werden, hängt von den anfallenden Kosten ab, und so kann es vorkommen, dass nur Sicherheitsdefizite korrigiert werden, die einen größeren Umsatz- oder Imageverlust zur Folge haben könnten.

Die Verfügbarkeit von Updates und das Einstellen des Produktsupportes liegen im Ermessen des Anbieters und können außerdem durch politische Interessen eingeschränkt werden. So möchte die US-Regierung über unbekanntes aber existierende Sicherheitsmängel informiert werden (Anderson, 2002), damit Geheimdienste und Starfverfolgungsbehörden die Schwachstelle so lange ausnutzen können bis diese von der Öffentlichkeit entdeckt wird, erst jetzt darf der Softwareentwickler die Sicherheitslücken schließen. Die beabsichtigten Sicherheitslücken bieten nicht nur bestimmten Organisationen Zugriff zum System, sondern können auch anderen Angreifern Zugang ermöglichen.

Da der Quellcode bei Open-Source-Software hingegen verfügbar ist, kann jeder gefundene Fehler theoretisch selbstständig, unabhängig von einem Hersteller, entfernt werden (Gehring, 2001). Oft werden Fehler von Benutzern gefunden und gemeldet, die aber dann von anderen Anwendern oder Entwicklern verstanden und womöglich wiederum von anderen behoben werden. Dieses Vorgehen ist parallelisierbar und lässt sich so ideal auf eine große, verteilte Gemeinde anwenden - ein ganz entscheidender Vorteil gegenüber der traditionellen Softwareentwicklung.

Durch den offenen Code ist es möglich, Software an die eigenen Sicherheitsvorstellungen anzupassen (Kamat, 2004). So können Organisationen mit hohen sicherheitstechnischen Ansprüchen ihre Software durch geeignete Verfahren härten, außerdem können die Sicherheitsaspekte der einzelnen Programme verglichen werden, so kann man sich bei der angebotenen Open-Source-Software für die sicherste Lösung entscheiden. Dadurch entsteht ein Wettbewerb, der sich positiv auf die gebotene Sicherheit auswirken dürfte.

5.4 Erleichtert die Veröffentlichung des Quellcode Angreifern Sicherheitslücken zu finden?

Um Sicherheitslücken zu finden, können Angreifer dynamische oder statische Vorgehensweisen wählen (Kamat, 2004). Mit dynamisch ist hiermit gemeint,

dass Untersuchungen am laufenden Programm vorgenommen werden, z.B. durch direkten Zugriff auf Speicherbereiche oder die Übergabe von problematischen Daten, um dann über die Programmreaktionen mögliche Sicherheitslücken herauszufinden.

Bei der statischen Methode wird der Code betrachtet und nach Mustern durchsucht, die auf Schwachstellen hindeuten könnten (Kamat, 2004). Bei Open-Source-Software wird hierzu der Quellcode genommen, während bei Closed-Source-Software der Maschinencode oder per Reverse Engineering¹² gewonnener Code durchsucht wird.

D.h., es besteht im dynamischen Fall kein Unterschied zwischen Open-Source- und Closed-Source-Software, während sich im statischen Fall bei Closed-Source-Software die Suche im Code schwieriger gestaltet als bei Open-Source-Software. Durch die zahlreichen Möglichkeiten zur Schwachstellensuche bei Closed-Source-Software besitzt die Geheimhaltung des Quelltextes keine nennenswerten Vorteile, bis vielleicht darauf, dass man einen Zeitvorteil besitzt, wenn noch nicht öffentlich bekannte Sicherheitslücken geschlossen werden sollen.

5.5 Vergleich von Windows- und Linuxdesign

5.5.1 Einzel- vs. Mehrnutzersystem

- Windows wurde als Einzelnutzersystem entwickelt, in welchem der Benutzer Administratorrechte besitzt. Sensible Bereiche des Betriebssystems sind sowohl dem Benutzer als auch den Programmen frei zugänglich, d.h. schädlicher Code wird mit administrativen Rechten ausgeführt (Anderson, 2001). Mit der Ausstattung des Benutzers mit begrenzten Systemprivilegien versuchte Microsoft den Nutzer- und Systembereich voneinander zu trennen und Windows zu einem Mehrbenutzersystem auszubauen (Petreley, 2004). Dies hatte zur Folge, dass alte Programme ihren Dienst verweigerten (Petreley, 2004), weil sie auf die Manipulation von Systemdateien angewiesen waren. Dem begegnete Microsoft mit der Einführung eines Kompatibilitätsmodus, welches den Programmen wieder - zu Lasten der Sicherheit - mehr Privilegien einräumte.
- Linux wurde von Grund auf als Mehrbenutzersystem entworfen, welches streng zwischen Benutzer und Systemdaten trennt und Nutzern Manipulationen, welche weitreichende Folgen fürs System haben, nur erlaubt, falls diese ausreichende Zugriffsrechte besitzen. Der Vorteil eines Mehrnutzersystems ist, dass schädliche Programme die gleichen Privilegien wie der Nutzer selbst besitzen und deshalb nur begrenzt Schaden verursachen können (Petreley, 2004).

5.5.2 Monolithische vs. modulare Architektur

- Bei der Windowsarchitektur werden mehrere funktionale Einheiten zu einer Einheit zusammengefasst (Petreley, 2004) - im Gegensatz zum mo-

¹²Reverse Engineering bezeichnet den Vorgang zur Rückgewinnung des Quellcodes oder einer vergleichbaren Beschreibung aus Binärcode, z.B. von einem ausführbaren Programm oder einer Programmbibliothek, etwa mit einem Disassembler oder einem Decompiler. In der Regel kann der komplette Programmquellcode nicht vollständig zurückgewonnen werden, da z.B. Kommentare und lokale Objektamen nur selten im verfügbaren Binärcode enthalten sind.

dularen Gedanken, in welchem wenige Funktionen in einzelnen Schichten zusammengefasst werden, wobei jede Schicht eingeschränkten Zugriff auf die anderen Schichten besitzt.

Ein Beispiel für den monolithischen Aufbau ist der Internet Explorer, welcher eng mit dem Betriebssystem verwoben ist. Viele Microsoftprodukte und Software von Drittanbietern machen von den Funktionalitäten des Internet Explorers Gebrauch, so dass eine Schwachstelle im Browser eine Gefahr für das ganze System werden kann.

Anstatt den Kernel nur mit den notwendigsten hardwarenahen Funktionalitäten auszustatten, integrierte Microsoft zu viele Features im Kernel - Instabilität und Sicherheitsmängel sind die Folge. Nachteilig an monolithisch aufgebauten Systemen sind auch die vielen Abhängigkeiten zwischen den einzelnen Softwarekomponenten anzusehen - wird ein Softwareteil ersetzt, resultieren weitreichende Folgen für das ganze System. Wird z.B. eine Systemschwachstelle durch Veränderung des Codes behoben, können unbeabsichtigte Seiteneffekte zu neuen Mängeln im System führen.

- Die Modularität von Linux ermöglicht weitestgehende Unabhängigkeit zwischen den einzelnen Komponenten, so dass sich Sicherheitslücken nicht auf andere Applikationen auswirken müssen. Jedoch sind nicht alle Systemkomponenten modular, so z.B. der Kernel (Petreley, 2004). Doch dem Nachteil einer nicht modularen Architektur wird mit einer strikten Kernelphilosophie begegnet: „Wann immer es möglich ist, eine Funktionalität außerhalb des Kernels zu implementieren, so muss diese Funktionalität außerhalb implementiert werden.“

5.5.3 Starke vs. schwache Bindung an ein RPC-Modell

- Mithilfe von RPC¹³ können Rechner über ein Netzwerk miteinander kommunizieren oder auch ferngesteuert werden. Um sich gegen Angriffe, die über RPC getätigt werden oder sich auf eine Schwachstelle von RPC beziehen, zu schützen, kann man die betreffenden Ports per Firewall sperren. Dies ist jedoch nicht immer möglich, da viele Windowsdienste starken Gebrauch von RPC machen. Nachteilig zu bewerten ist, dass RPC bei Bedarf nicht vollständig deaktiviert werden kann.

Ein Beispiel für die exzessive Nutzung des RPC ist der Microsoft SQL-Server (Petreley, 2004), bei welchem die Möglichkeit besteht mehrere Serverinstanzen auf einem Rechner zu starten. Die Kommunikation findet dabei über RPC statt. Dies ist nicht nur überflüssig - die einzelnen Instanzen befinden sich auf dem gleichen Rechner, hätten also direkt miteinander Daten austauschen können - sondern auch sicherheitstechnisch bedenklich.

- Die meisten Linuxapplikationen haben per Default-Einstellung keinen Netzwerkzugriff. Installiert man MySQL und den Apache-Server auf dem gleichen Rechner, um eine Webseite ins Internet zu stellen, so benötigt MySQL

¹³Remote Procedure Call (RPC) ist ein Netzwerkprotokoll auf der fünften und sechsten Schicht des ISO/OSI-Modells. Mit Hilfe von RPC können über ein Netzwerk Funktionsaufrufe auf entfernten Rechnern durchgeführt werden. RPC wurde ursprünglich durch Sun Microsystems für NFS entwickelt. Der genaue Aufbau von RPC wird in den RFCs 1057 und 1831 beschrieben.

keinen Zugriff auf das Netzwerk, um mit dem Apache-Server zu interagieren. Und selbst wenn eine Netzwerkverbindung notwendig wäre, kann der SQL-Server so konfiguriert werden, dass er Remoteaufrufversuche nur von speziellen vorher in der Konfigurationsdatei eingetragenen Rechnern erlaubt. Im Gegensatz zu Windows können alle RPC Dienste deaktiviert werden, ohne die Funktionsfähigkeit des Systems einzuschränken (Petreley, 2004).

5.6 Sicherheitsvergleich bei Webservern in der Praxis

Ähnliche Designentscheidungen wie bei Windows lassen sich auch bei Microsofts Internet Information Server (IIS) finden. So ist der Webserver monolithisch aufgebaut, sichert dem Nutzer hohe Privilegien zu und besitzt mehrere sicherheitstechnisch bedenkliche Default-Einstellungen. Der Apache-Webserver hingegen besitzt so wie Linux einen modularen Aufbau und stattet den Benutzer je nach Konfiguration mit geringen bis hohen Nutzungsrechten aus.

Man könnte denken, dass, wenn Linux genauso weit verbreitet wäre wie Windows, es mit den gleichen Sicherheitsproblemen zu kämpfen hätte. Aufgrund der großen Anzahl von Windows-Systemen werden diese bevorzugt Ziel schädlicher Programme, während die kleine Menge von Linuxrechnern keine lohnende Angriffsfläche bietet. Dass Windows eine weite Verbreitung im Desktopbereich besitzt, ist unbestritten. Daraus muss jedoch nicht folgen, dass Windows und Linux bei gleich starker Verteilung ein ähnliches Maß an Sicherheit bieten würden (Kamat, 2004). Würde man das Argument auf den Apache-Server anwenden, der mit einem Marktanteil von 68% weiter verbreitet ist als die Microsoft Alternative IIS mit 21%, so würde daraus folgen, dass für den Apache mehr schädliche Programme existieren müssten. Dies ist jedoch nicht der Fall. Der Apache-Server zeichnet sich durch seine hohe Sicherheit und Stabilität aus (Ritchey u. a., 2001), während der IIS bevorzugtes Ziel von Angriffen darstellt (Code Red, Red.A, IISWorm, etc.).

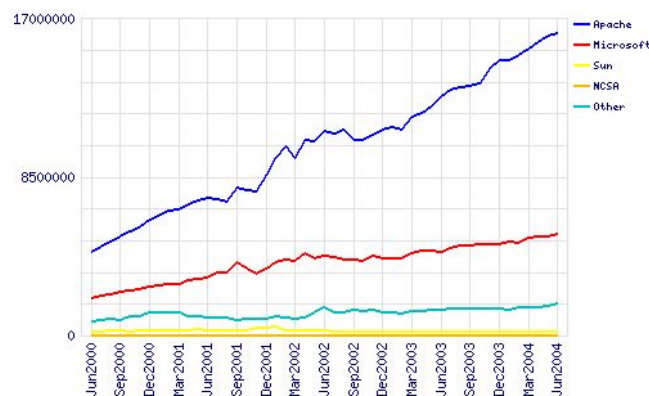


Abbildung 5: Marktanteile von Webservern (Wheeler, 2004)

Über den Zeitraum von 1996 bis 2001 wurden die beiden populärsten Webserver, der Apache-Server aus dem Open-Source-Bereich und der IIS aus dem Closed-Source-Bereich, sicherheitstechnisch miteinander verglichen (Ritchey u. a.,

2001). Als Vergleichskriterien wurden zum einen die Zeit von dem Bekanntwerden einer Sicherheitslücke bis zur Behebung und zum anderen die Anzahl der aufgetretenen Schwachstellen benutzt. Desweiteren wurden die einzelnen Fehler ihrer Schwere nach in drei Kategorien eingeteilt, wobei die Kategorie eins für die schwersten Fehler, wie die Möglichkeit auf dem Rechner Zugriff mit administrativen Rechten zu erhalten, und die Kategorie drei für leichtere Sicherheitsmängel steht.

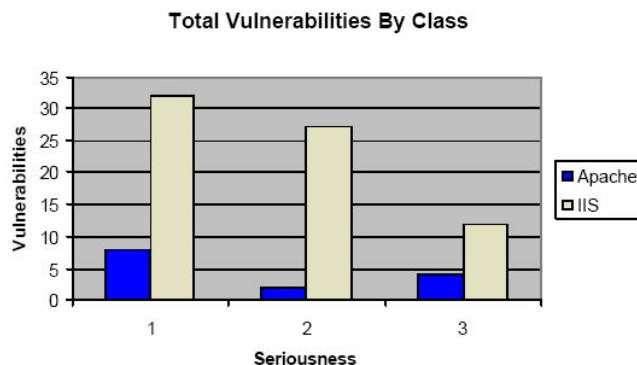


Abbildung 6: Anzahl der Sicherheitsmängel nach Schwere klassifiziert (Ritchey u. a., 2001)

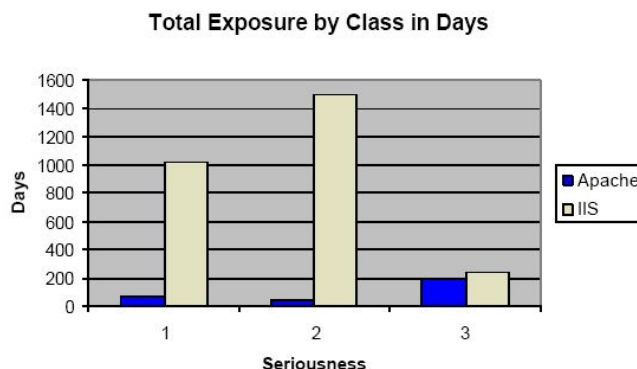


Abbildung 7: Durchschnittliche Anzahl der Tage bis zur Behebung der Sicherheitsmängel (Ritchey u. a., 2001)

Aus den Diagrammen ist die sicherheitstechnische Überlegenheit des Apache-Servers ersichtlich. So besitzt der Apache viermal weniger schwerwiegende Fehler der Klasse eins als der IIS, und auch in der Fehlerklasse zwei ist er überlegen.

Die Reaktionszeit auf schwerwiegende Fehler der Klassen eins und zwei ist beim Apache-Server sehr viel kürzer. Nur bei der Behebung von weniger kritischen Sicherheitslücken sind die Reaktionszeiten beider Webserver fast identisch.

Ähnliche Ergebnisse zu Gunsten des Open-Source-Programms hatte eine Studie, die die Sicherheit von Linux und Windows untersuchte (Ritchey u. a., 2001).

5.7 Fazit

Allein der Umstand, dass der Code veröffentlicht wird, macht die Software nicht sicherer. Open Source bietet jedoch eine vielversprechende Grundlage für Code, der hohen Sicherheitsansprüchen genügen soll.

Viele Methoden, wie die Wiederverwendung von Code, die in kurzen Zeitabständen erfolgenden Releases und die Nähe zwischen Entwickler und Anwender begünstigen die Entwicklung von sicherer Software. Hinzu kommt, dass die Entwickler von Open-Source-Software im Gegensatz zu den meisten Closed-Source-Projekten unabhängig vom Marktdruck und von ökonomischen Interessen sind.

Herstellerunabhängigkeit, Transparenz, die Möglichkeit zur eigenständigen Fehlerbehebung und die Anpassung der Software an die eigenen Sicherheitsansprüche wären die Vorteile auf Seiten des Anwenders.

Die sicherheitstechnische Qualität von Linux und dem Apache-Webserver kann vielleicht nicht auf alle Open-Source-Programme verallgemeinert werden, stellt jedoch ein Indiz für den hohen Sicherheitsanspruch bei Open-Source-Projekten dar.

Literatur

- [Anderson 2001] ANDERSON, Ross: *Why Information Security is Hard - An Economic Perspective*. Presented at the applications security conference. 12 2001. – URL <http://www.cl.cam.ac.uk/ftp/users/rja14/econ.pdf>
- [Anderson 2002] ANDERSON, Ross: *Security in Open versus Closed Systems - The Dance of Boltzmann, Coase and Moore*. Open Source Software: Economics, Law and Policy Conference in Toulouse (France). 6 2002. – URL <http://www.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf>
- [Barr 2002] BARR, Joe: *LinuxWorld: Seeing Through the Linux-Windows TCO Comparisons*. http://linuxtoday.com/news_story.php3?ltsn=2002-12-21-007-26-OP-BZ-DP, 21. Dezember 2002
- [Bielecke 2002] BIELECKE, J.: *Münchens Software heißt Linux*. Süddeutsche Zeitung, 27. Mai 2002
- [Braeuner 2004] BRAEUNER, H.: *Linux im Rathaus - Update 2004*. Dornbirn : LinuxKongreß 2004, 12. November 2004
- [Challet u. a. 2004] CHALLET, Damien u. a.: Microscopic model of software bug dynamics: closed source versus open source. In: *International Journal of Reliability, Quality and Safety Engineering* (2004), 10
- [Dieckheuer und Kooths 2003] DIECKHEUER, Gustav (Hrsg.) ; KOOTHS, Stefan (Hrsg.) ; Muenster Institute for Computational Economics, Universität Münster (Veranst.): *Open Source-Software - Eine volkswirtschaftliche Betrachtung*. Dezember 2003. (MICE Economic Research Studies 4)
- [Free Software Foundation 2004] FREE SOFTWARE FOUNDATION: *Philosophy of the GNU Project*. <http://www.gnu.org/philosophy>, 2004
- [Gehring 2001] GEHRING, Robert A.: *Unsichere Software - Eine systemische Betrachtung*. Vortrag auf der Herbsttagung der Europäischen Akademie Bad Neuenahr-Ahrweiler. 11 2001
- [Gehring u. a. 2004] GEHRING, Robert A. u. a.: *Open Source Jahrbuch 2004 - Zwischen Softwareentwicklung und Gesellschaftsmodell*. Lehmanns Media, 2004
- [Ghosh u. a. 2002] GHOSH, Rishab A. ; KRIEGER, Bernhard ; GLOTT, Ruediger ; ROBLES, Gregorio: Part 2B: Open Source Software in the Public Sector: Policy within the European Union. In: *Free/Libre and Open Source Software: Survey and Study*. Universität Maastricht, Juni 2002
- [Hilbert 2002] HILBERT, T.: *Schwäbisch Hall setzt komplett auf Linux*. Pressemitteilung der Stadt Schwäbisch Hall, 27. November 2002
- [Hilbert 2003] HILBERT, T.: *Linux in Schwäbisch Hall: Glückwunsch von Schily*. Pressemitteilung der Stadt Schwäbisch Hall, 14. März 2003
- [Hoegner 2004] HOEGNER, W.: *Das Projekt LiMux stellt sich vor*. E-Government-Forum, Systems 2004, 19. Oktober 2004

- [Jones 2004] JONES, Pamela: *GROKLAW*. <http://www.groklaw.net>, 2004
- [Kamat 2004] KAMAT, Pandurang: Application Security : The case for Open Source Software / Rutgers University - Department of Computer Science. URL <http://paul.rutgers.edu/pkamat/>, 2004. – Forschungsbericht
- [KBSt 2003] Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern (Veranst.): *Migrationsleitfaden*. Juli 2003. (Schriftenreihe der KBSt 57)
- [Kenwood 2001] KENWOOD, Carolyn A.: *A Business Case Study of Open Source Software*. MITRE Washington C3 Center, Bedford, Massachusetts : The MITRE Corporation, Juli 2001
- [Kloiber 2003] KLOIBER, M.: *Gestatten, mein Name ist Tux*. Welt am Sonntag, 05. November 2003
- [Leiteritz 2002] LEITERITZ, Raphael: *Der kommerzielle Einsatz von Open Source Software und kommerzielle Open Source-Geschäftsmodelle*, Technische Universität Berlin, Diplomarbeit, 2002
- [Lessig 2002] LESSIG, Lawrence: Open Source Baselines: Compared to What? In: HAHN, Robert W. (Hrsg.): *Government Policy toward Open Source Software*. Washington, D.C. : AEI-Brookings Joint Center for Regulatory Studies, 2002
- [MS 2004] MICROSOFT COOPERATION: *Get the Facts Home*. <http://www.microsoft.com/getthefacts>, 2004
- [München 2004] MÜNCHEN, Direktorium der Landeshauptstadt: *Das Projekt: „LiMux - Die IT-Evolution“*. <http://www.muenchen.de/Rathaus/dir/limux/89256/index.html>, 2004
- [Näve Leuchten GmbH 2004] NÄVE LEUCHTEN GMBH: *Informationsseiten*. <http://www.naeve.de>, 2004
- [Bayrischer Oberster Rechnungshof 2001] OBERSTER RECHNUNGSHOF Bayerischer: *Jahresbericht 2001*. <http://www.orh.bayern.de/Jahresbericht2001.pdf> : Bayerischer Oberster Rechnungshof, München, 2001
- [Open Source Initiative 2004] OPEN SOURCE INITIATIVE: *The Open Source Definition*. http://www.opensource.org/docs/definition_plain.php, 2004
- [Open Source Risk Management 2004] OPEN SOURCE RISK MANAGEMENT: *Preserving the Benefits of Open Source & Protecting Users Against Risk*. <http://www.osriskmanagement.com>, 2004
- [Petreley 2004] PETRELEY, Nicholas: Security Report: Windows vs. Linux. In: *The Register* (2004), 10. – URL http://www.theregister.co.uk/security/security_report_windows_vs_linux/
- [Raymond 1997] RAYMOND, Eric S.: *The Cathedral and the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 1997. – URL <http://catb.org/esr/writings/cathedral-bazaar/>

- [Ritchey u. a. 2001] RITCHEY, Ronald W. u. a.: Open Source Vs. Close Source Software - An Experiment To Determine Which is More Secure. URL <http://www.isse.gmu.edu/faculty/ofut/classes/763/studpapers/Ritchey-1106.pdf>, 2001. – Forschungsbericht
- [Sedlmayer und Gigerich 2004] SEDLMAYER, Roman ; GIGERICH, Jan: *Rechtliche Bedingungen und Risiken der Landeshauptstadt München für den Einsatz von Open-Source Software*. Kanzlei Frohwitter, Online: <http://www.rismuenchen.de/RII/RII/DOK/SITZUNGSVORLAGE/517379.pdf>, 10. September 2004
- [Snoopy und Müller 1999] SNOOPY ; MÜLLER, Martin: *Open Source - kurz & gut*. O'Reilly & Associates, April 1999
- [Strobl 2004] STROBL, F.: *Warum Open Source - Die Hintergründe der Münchener Entscheidung*. E-Government-Forum, Systems 2004, 19. Oktober 2004
- [SUN 2004] SUN MICROSYSTEMS: *TCO Calculator for Sun Java System Communications Software*. http://www.sun.com/software/products/messaging_srvr/tco/tco.jsp, 2004
- [Unilog Integrata 2003] UNILOG INTEGRATA: *Client Studie der Landeshauptstadt München - Kurzfassung des Abschlußberichts inklusive Nachtrag*. http://www.muenchen.de/vip8/prod1/mde/_de/rubriken/Rathaus/40_dir/linux/publikationen/clientstudie.kurz.pdf, 02. Juli 2003
- [Varian und Shapiro 2003] VARIAN, Hal R. ; SHAPIRO, Carl: *Linux Adoption in the Public Sector: An Economic Analysis*. Online: <http://www.sims.berkeley.edu/~hal/Papers/2004/linux-adoption-in-the-public-sector.pdf> : University of California, Berkeley, 2003
- [Wheeler 2004] WHEELER, David A.: *Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!* (2004), 11. – URL http://www.dwheeler.com/oss_fs_why.html
- [Wieland 2004] WIELAND, Thomas: *Stärken und Schwächen freier und Open-Source-Software im Unternehmen*. In: GEHRING, Robert A. (Hrsg.) ; LUTTERBECK, Bernd (Hrsg.): *Open Source Jahrbuch 2004 - Zwischen Softwareentwicklung und Gesellschaftsmodell*. Lehmanns Media - LOB.de, Berlin, 2004
- [Wild und Herges 2000] WILD, Martin ; HERGES, Sascha: *Total Cost of Ownership (TCO) - Ein Überblick*. In: WIRTSCHAFTSINFORMATIK, Lehrstuhl für Allg. BWL und (Hrsg.): *Arbeitspapiere WI Nr. 1/2000*. Mainz : Johannes Gutenberg-Universität, 2000