2.2004 Schriftenreihe

Der Österreichische Städtebund informiert.

Open Source Software – Einsatz in der öffentlichen Verwaltung

ÖSTERREICHISCHER STÄDTEBUND

Open Source Software – Einsatz in der öffentlichen Verwaltung

Emil Georgiev
Gottfried Haber
Julia Reifensteiner
Ronald Sallmann

Schriftenreihe des Österreichischen Städtebundes
Open Source Software-Einsatz in der öffentlichen Verwaltung

Herausgeber:
Österreichischer Städtebund
1082 Wien, Rathaus
Telefon: 01/4000-89980
Telefax: 01/4000-7135
E-Mail: post@stb.or.at
Internet: http://www.staedtebund.at

Schriftleitung:
Generalsekretär Dkfm. Dr. Erich Pramböck

Koordination und Betreuung:
Dr. Ronald Sallmann

Autoren:

Kapitel 1:	Grundlagen zu Open Source Software	Julia Reifensteiner
Kapitel 2:	SWOT – Betrachtung eines Open Source Software-Einsatzes in der öffentlichen Verwaltung	Ronald Sallmann
Kapitel 3:	Kritische Erfolgsfaktoren und Umstiegsszenarien	Ronald Sallmann Julia Reifensteiner
Kapitel 4:	Wirtschaftliche Aspekte von Open Source Software in der öffentlichen Verwaltung	Gottfried Haber
Kapitel 5:	Rechtliche Aspekte eines Open Source Software-Einsatzes in der öffentlichen Verwaltung	Emil Georgiev (Leitung: Wolfgang Zankl)

(e-center
vienna | brussels | london | leipzig | prague | budapest
europäisches zentrum für e-commerce und internetrecht
european center for e-commerce and internet law

Umschlaggestaltung und DTP-Produktion:
Karin Wieser, Grafic & Design
1030 Wien, Marokkanergasse 16
www.grafic.at

Druck:
Széchenyi Nyomda, H-9024 Győr

Wien, Oktober 2004

Vorwort

Die vorliegende Ausgabe der Schriftenreihe „Der Österreichische Städtebund informiert“ widmet sich einem Themengebiet, das in den Städten und Gemeinden, aber auch in der gesamten öffentlichen Verwaltung, verstärkt für Aufmerksamkeit sorgt, nämlich dem Einsatz von freier Software in der Verwaltung. Auch die mediale Beachtung ist groß, gibt eine Regierung eine eindeutige Absichtserklärung ab, in Zukunft verstärkt auf Open Source Software zu setzen, oder entschließt sich eine Stadt zum Umstieg.

Vor dem Hintergrund steigender Ausgaben für Informations- und Telekommunikationstechnologien auf der einen Seite und dramatisch schrumpfender Budgets auf der anderen Seite sucht nun vermehrt auch die öffentliche Hand nach Alternativen. Freie Software, die kostengünstig – teilweise sogar kostenlos – verfügbar ist, scheint auf den ersten Blick als die ideale Problemlösung. Doch die Komplexität der öffentlichen Verwaltung ist hoch und die zur Bearbeitung der vielfältigen Aufgaben eingesetzte Software vielfältig. Hinzu kommen die besondere Sensibilität der behördlichen Tätigkeit, die Rechtsverbindlichkeit und der hohe Anspruch an den Schutz der anvertrauten Daten. Nicht zu vergessen die Leistungsverpflichtung der öffentlichen Hand: Für ein kommerzielles Unternehmen bedeuten Ausfälle oder gar Stillstand der EDV-Systeme Einnahmenausfälle, für die öffentliche Hand würde es hingegen bedeuten, garantierten Leistungen, von denen die Leistungsempfänger in vielerlei Hinsicht abhängig sind, nicht nachkommen zu können. Somit steht auch die Zuverlässigkeit und Ausfallssicherheit elektronischer Systeme ganz oben auf der Anforderungsliste für Software.

An einen Umstieg auf freie Software knüpfen sich daher eine lange Reihe von Überlegungen und Abwägungen, die zu bedenken sind. Und es gibt kein „Patentrezept“ für einen Umstieg, vielmehr verschiedene Szenarien vor dem Hintergrund der individuellen Situation und IT-Ausstattung jeder einzelnen Gebietskörperschaft.

Die hohe Aufmerksamkeit für das Thema in Verbindung mit den vielen Unsicherheiten und Facetten, die ein Einsatz von Open Source Software in der öffentlichen Verwaltung mit sich bringt, waren schließlich auch Anlass für den Österreichischen Städtebund, dieses Thema in einer eigenen Publikation aufzubereiten. Der vorliegende Band versteht sich als Basisinformation zum Einstieg in die sehr komplexe und technische Materie, wobei versucht wurde, die wesentlichen Aspekte objektiv und möglichst wenig technologielastig darzustellen. Aufgrund der vorrangigen Bedeutung für die öffentliche Verwaltung widmen sich zwei Kapitel darüber hinaus der wirtschaftlichen und der rechtlichen Perspektive eines Open Source Software-Einsatzes in der öffentlichen Verwaltung.

Auch vor dem Hintergrund der E-Government-Entwicklung gewinnt das Thema „Open Source“ an zusätzlicher Bedeutung, da E-Government-Anwendungen zwecks Verbreitung vermehrt unter freie Lizenz gestellt werden und der Einsatz von offenen Standards empfohlen wird. Damit schließt sich auch der Kreis, weshalb sich der Österreichische Städtebund diesem Spezialthema im Rahmen seiner E-Government-Initiative widmet.

Wien, im Oktober 2004

Dkfm. Dr. Erich Pramböck
Generalsekretär

INHALTSVERZEICHNIS

EXECUTIVE SUMMARY

Der Einsatz von so genannter „Open Source Software“ wird zunehmend auch in der öffentlichen Verwaltung zu einem Thema. Es handelt sich dabei um eine komplett gegensätzliche Weltanschauung zu herkömmlicher Software: „Open Source“ bedeutet, dass der Quellcode – also die eigentliche Grundlage jeder Software – frei verfügbar und veränderbar ist. Die Entwicklung von Open Source Software erfolgt vielfach auf private Initiative ambitionierter Programmierer, die in ihrer Freizeit an derartigen Projekten arbeiten. Auf dieser Grundlage hat sich eine weltumspannende „Community“ gebildet und Produkte hervorgebracht, die zunehmend in direkter Konkurrenz zu kommerziell hergestellter Software stehen. Das „Leitprodukt“ der Open- Source-Welt ist GNU/Linux, ein modular aufgebautes Betriebssystem, für das es unzählige Erweiterungen gibt.

Das Bestechende an der Open-Source-Idee ist die Möglichkeit, Programme einzusehen, zu verändern und weiterzugeben, ohne rechtliche Konsequenzen fürchten zu müssen. Dafür sollen eigens geschaffene Lizenzen garantieren, unter die Open Source Software gestellt wird. Es gibt unterschiedliche Lizenzmodelle, die verschiedene Grade der Modifizierung, der Weitergabe und auch der kommerziellen Nutzung von Open Source Software kennen. Die populärste und auch am häufigsten verwendete Lizenz ist GNU-GPL, die GNU General Public Licence.

Der frei verfügbare Quellcode ist gleichzeitig auch der „kleinste gemeinsame Nenner“ der Open-Source-Welt: Nicht alle unter Open-Source-Lizenz gestellten Produkte sind kostenlos und die Entwicklung wird nicht ausschließlich von Hobbyprogrammierern betrieben. Es gibt auch bei Open Source Software verschiedene Geschäftsmodelle. Manche Unternehmen verdienen an der Zusammenstellung anwendungsoptimierter Open Source Software-Pakete, andere Unternehmen an der Implementierung, und es geschieht auch Auftragsentwicklung auf Open-Source-Basis.

Es wäre auch ein Irrtum anzunehmen, dass die Open-Source-Community ein anarchisch agierendes oder unorganisiertes Netzwerk sei. Über die Vergabe und Einhaltung der Open-Source-Lizenzen, über die Weiterentwicklung von Leitprojekten und über die Einhaltung bestimmter Regeln wachen Organisationen wie die „Free Software Foundation“, die „Open-Source-Initiative“ oder das „Free Software Consortium“. Innerhalb von Open Source Software-Projekten gibt es ebenfalls klare Regeln und Hierarchien, um nach Möglichkeit zu verhindern, dass sich ein Produkt undifferenziert entwickelt, dass – vorsätzlich oder unabsichtlich – Fehler auftreten oder dass sich ein Projekt aufgrund von Uneinigkeiten im Entwicklerteam sogar aufspaltet.

Der öffentlichen Verwaltung bietet Open Source Software eine interessante Alternative zu kommerzieller – meist proprietärer – Software: Es fallen keine Lizenzgebühren an, Open Source Software ist im Ankauf wesentlich günstiger, teilweise sogar kostenlos, und da der Quellcode frei verfügbar ist, können notwendige oder erwünschte Änderungen im Prinzip von jedem dazu fähigen Entwickler vorgenommen werden. Es besteht keinerlei Abhängigkeitsverhältnis von einem Hersteller. Doch der Einsatz von Open Source Software birgt auch eine Reihe von Stolpersteinen: Die Hardwareunterstützung ist deutlich schlechter als bei proprietärer Software, sodass bei einem Umstieg mit Problemen bei der bestehenden EDV-Ausstattung gerechnet werden muss. Nicht alle benötigten Produkte – und vor allem Fachanwendungen für die öffentliche Verwaltung – sind auf Open-Source-Basis verfügbar, was einen aufwendigeren Mischbetrieb von proprietärer Software

und Open Source Software vonnöten macht. Bei Fehlern kann darüber hinaus auch kein Hersteller zur Verantwortung gezogen werden, es sei denn, es handelt sich um eine Auftragsentwicklung.

Aus den Vor- und Nachteilen von Open Source Software resultieren auch eine Reihe von Chancen bzw. Risiken: Die Anbieterunabhängigkeit erlaubt marktgerechte Preise und einen Wechsel des Supportgebers, außerdem ist es eher möglich, die lokale bzw. nationale Wirtschaft zu fördern. Verwaltungsübergreifende Entwicklungsgemeinschaften wären möglich und könnten dazu beitragen, die Entwicklungskosten für jeden Partner niedrig zu halten.

Auf der Risikoseite steht hingegen die Gefahr eines hohen Migrationsaufwandes, unvorhergesehener Probleme und damit auch Kosten und Verzögerungen, gewisse rechtliche Unsicherheiten beim Einsatz von Open Source Software und eine neue Form der Abhängigkeit, nämlich nicht mehr vom Hersteller, sondern vielmehr vom Supportgeber. Nicht vergessen werden darf auf den Faktor „Personal“, wird doch bei einem Umstieg auf Open Source Software – zumindest im Arbeitsplatzumfeld – unmittelbar in tägliche Abläufe und Routinen eingegriffen.

Die Komplexität eines Open-Source-Umstellungsprojekts darf nicht unterschätzt werden. Neben einem individuellen Abwägen der Stärken, Schwächen, Chancen und Risiken durch jede veränderungswillige Gebietskörperschaft muss ein derartiges Projekt auch hundertprozentig von der Verwaltungsführung getragen werden. Das Um und Auf für das Gelingen ist – mehr noch als bei anderen Projekten – eine umfassende Analyse der Anforderungen und der Ausgangssituation, eine präzise Projektplanung und schlussendlich auch eine konsequente Umsetzung, begleitet von umfassender Mitarbeiterinformation und von Schulungen.

Der Einsatz von Open Source Software kennt viele Aspekte, für die öffentliche Verwaltung sind jedoch zwei Aspekte von vorrangiger Bedeutung, nämlich die wirtschaftliche und die rechtliche Betrachtung, weshalb ihnen in der vorliegenden Publikation besonderer Raum gewidmet wird.

In die ökonomischen Betrachtungen sind die grundsätzlichen Unterschiede zwischen proprietärer und Open Source Software mit einzubeziehen. Für umstiegswillige Gebietskörperschaften ist vor allem die Kostendimension ein entscheidender Faktor. Hier gibt es bereits verschiedene Modelle am Markt, nach denen die TCO – die Total Costs of Ownership – berechnet werden können.

Bei der rechtlichen Betrachtung sind vor allem zwei Faktoren ausschlaggebend: dass nach einem Umstieg auf Open Source Software nachträglich Rechtsunsicherheiten auftreten, deren Auflösung mit hohen Kosten oder einem erneuten Softwarewechsel verbunden sind. Der zweite Risikofaktor sind Sicherheitsbedenken, da die öffentliche Hand aufgrund ihrer staatlichen und gesetzlich legitimierten Tätigkeit besonders hohe Ansprüche an die Sicherheit von IT-Systemen hat.

Trotz aller Für und Wider eröffnet Open Source Software der öffentlichen Hand in jedem Fall eine interessante Entscheidungsoption im IT-Bereich – und sei es nur als Korrektiv für die Produkt- oder Preisgestaltung von Anbietern kommerzieller, proprietärer Software. Auf allen staatlichen Ebenen und auch auf europäischer Ebene sind Open-Source-Projekte im Laufen, sodass sich schon in naher Zukunft zeigen wird, wie gut Open Source Software tatsächlich in der öffentlichen Verwaltung einsetzbar ist.

Kapitel 1

GRUNDLAGEN ZU OPEN SOURCE SOFTWARE

1.1 EINLEITUNG – WAS VERSTEHT MAN UNTER OPEN SOURCE SOFTWARE?

Die Bezeichnung Open Source („Quelloffen“) Software bzw. Free Software wird für Programme verwendet, bei denen der User freien Zugang zum Quellcode[1] besitzt. Der Begriff „Freie Software“ versteht die Freiheit des Users, die Software nicht nur zu benutzen, sondern diese auch zu kopieren, weiterzugeben, zu verändern und zu verbessern[2], und nicht etwa den Preis der Software.

Definition von Open Source Software

Eine allgemein anerkannte Definition von Open Source Software stammt von der Open-Source-Initiative (OSI)[3]. Darin wird festgelegt, dass Open Source nicht nur freier Zugang zum Quellcode bedeutet, sondern dass eine Software, die unter dieser Bezeichnung vertrieben wird, darüber hinaus folgende Kriterien aufweisen muss[4]:

I Freie Weitergabe

Die Lizenz darf niemanden in seinem Recht einschränken, die Software als Teil eines Softwarepaketes, das Programme unterschiedlichen Ursprungs enthält, zu verschenken oder zu verkaufen. Die Lizenz darf für den Fall eines solchen Verkaufs keine Lizenz- oder sonstigen Gebühren festschreiben.

II Quellcode

Das Programm muss den Quellcode beinhalten. Die Weitergabe muss sowohl für den Quellcode als auch für die kompilierte[5] Form zulässig sein. Wenn das Programm in irgendeiner Form ohne Quellcode weitergegeben wird, so muss es eine allgemein bekannte Möglichkeit geben, den Quellcode zum Selbstkostenpreis zu bekommen, vorzugsweise als gebührenfreien Download aus dem Internet. Der Quellcode soll in einer Form vorliegen, die eine Bearbeitung durch einen Programmierer erlaubt. Absichtlich unverständlich geschriebener Quellcode ist daher nicht zulässig. Zwischenformen des Codes, so wie sie etwa ein Konverter („Translator”) erzeugt, sind unzulässig.

III Abgeleitete Software

Die Lizenz muss Veränderungen und Derivate zulassen. Außerdem muss sie es zulassen, dass die solcherart entstandenen Programme unter denselben Lizenzbestimmungen weitervertrieben werden können wie die Ausgangssoftware.

---

[1] Ein Quellcode ist ein in einer Programmiersprache geschriebener Zeilencode, die Grundlage einer Software. Meist ist er eine Sammlung von Dateien, manchmal ist der Quellcode auf Papier bzw. auf Band aufgenommen erhältlich. Änderungen der Software sind ausschließlich durch die Vornahme von Änderungen im Quellcode möglich!

[2] Vgl. http://www.gnu.org/philosophy/free-sw.html.

[3] Siehe Kapitel 1.2.4.

[4] Die vorliegende Open-Source-Definition ist eine Übersetzung des Fachverbands Unternehmensberatung und Informationstechnologie (UBIT) der Wirtschaftskammer Österreich, online unter: http://www.opensource.co.at/content.php, die Originalversion der Open-Source-Definition ist als Anhang eingefügt bzw. online abrufbar unter: www.opensource.org/docs/definition.php.

[5] Ein Kompilierer ist ein Computerprogramm, das ein im Quellcode geschriebenes Programm in ein Programm, das in der Zielsprache geschrieben wurde, übersetzt. Die kompilierte Form des Quellcodes ist somit der so genannte Zielcode oder Objektcode.

IV Unversehrtheit des Quellcodes des Autors
Die Lizenz darf die Möglichkeit, den Quellcode in veränderter Form weiterzugeben, nur dann einschränken, wenn sie vorsieht, dass zusammen mit dem Quellcode so genannte „Patch files" weitergegeben werden dürfen, die den Programmcode bei der Kompilierung verändern. Die Lizenz muss die Weitergabe von Software, die aus verändertem Quellcode entstanden ist, ausdrücklich erlauben. Die Lizenz kann verlangen, dass die abgeleiteten Programme einen anderen Namen oder eine andere Versionsnummer als die Ausgangssoftware tragen.

V Keine Diskriminierung von Personen oder Gruppen
Die Lizenz darf niemanden benachteiligen.

VI Keine Einschränkungen bezüglich des Einsatzfeldes
Die Lizenz darf niemanden daran hindern, das Programm in einem bestimmten Bereich einzusetzen. Beispielsweise darf sie den Einsatz des Programms in einem Unternehmen oder in der Genforschung nicht ausschließen.

VII Weitergabe der Lizenz
Die Rechte an einem Programm müssen auf alle Personen übergehen, die diese Software erhalten, ohne dass für diese die Notwendigkeit bestünde, eine eigene, zusätzliche Lizenz zu erwerben.

VIII Die Lizenz darf nicht auf ein bestimmtes Produktpaket beschränkt sein
Die Rechte an dem Programm dürfen nicht davon abhängig sein, ob das Programm Teil eines bestimmten Softwarepaketes ist. Wenn das Programm aus dem Paket herausgenommen und im Rahmen der zu diesem Programm gehörenden Lizenz benutzt oder weitergegeben wird, so sollen alle Personen, die dieses Programm dann erhalten, alle Rechte daran haben, die auch in Verbindung mit dem ursprünglichen Softwarepaket gewährt wurden.

IX Die Lizenz darf die Weitergabe zusammen mit anderer Software nicht einschränken
Die Lizenz darf keine Einschränkungen enthalten bezüglich anderer Software, die zusammen mit der lizenzierten Software weitergegeben wird. So darf die Lizenz z. B. nicht verlangen, dass alle anderen Programme, die auf dem gleichen Medium weitergegeben werden, auch quelloffen sein müssen.

Open Source Software entsteht durch die Zusammenarbeit einer Vielzahl von IT-Experten und Programmierern, die Interesse daran haben, hochwertige Software gemeinsam zu entwickeln, laufend zu verbessern, ihr Wissen und ihre Ergebnisse allgemein zugänglich zu machen und jedermann die Mitarbeit und Teilnahme am Entwicklungsprozess zu ermöglichen.

Die Entwickler von Open Source Software sehen ihr Wissen als Gemeingut, während – im Gegensatz dazu – Hersteller und Vertreiber von proprietärer Software[6] bestrebt sind, den Schutz ihres geistigen Eigentums ständig zu verbessern und auszuweiten[7].

[6] Proprietäre Software wird von Softwareherstellern aus kommerziellen Zwecken entwickelt. Der Quellcode ist nicht frei zugänglich und die Verwendung der Software unterliegt strengen lizenzrechtlichen Bestimmungen.
[7] Vgl. Grassmuck, 2000, S. 4.

Open Source Software wird von immer mehr Anwendern weltweit genutzt, da sie durch ihren offen gelegten Quellcode als zuverlässiger und sicherer als proprietäre Software gilt (z. B. Linux). Im Gegensatz zu proprietärer Software muss der User auch keine Lizenzgebühren zahlen und ist an keine restriktiven Lizenzmodelle gebunden. Dazu kommt, dass auch seitens der Europäischen Union der Einsatz von Open Source Software empfohlen und auch gefördert wird[8].

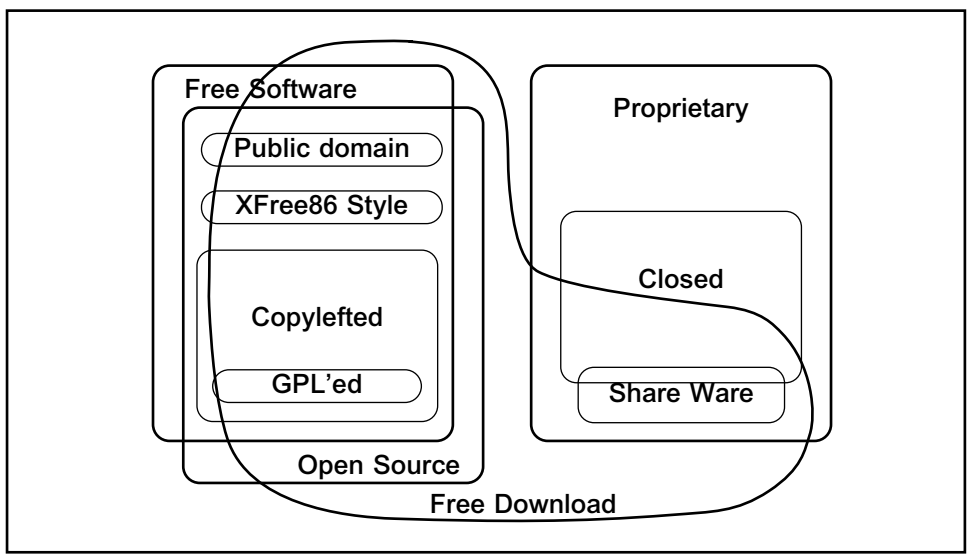
Auch für die öffentliche Verwaltung sind eine Reihe von Vorteilen bei der Verwendung von Open Source Software erkennbar. Aus diesem Grund wächst auch das Interesse der öffentlichen Verwaltung an Open Source Software, wobei diese in einigen europäischen Ländern als strategisches Instrument zur Verwaltungsmodernisierung und E-Government gefördert wird[9].

Varianten von Open Source Software

FLOSS ist das Akronym für „free, libre, open source software“ und steht für nicht proprietäre Software. Die Free Software Foundation (FSF) verwendet dafür ausschließlich die Bezeichnung „Free Software“, hingegen wird von der Open-Source-Initiative (OSI) die Terminologie „open source“ präferiert. Ebenso wird „libre“ verwendet, um freie, quelloffene Software zu umschreiben. Es handelt sich hierbei um terminologische Unterschiede, die eher ideologisch als technisch bedingt sind[10].

Im Zusammenhang mit Open Source werden häufig verschiedene Softwarekategorien verwendet. Diese greifen teilweise ineinander oder sind Teile anderer Kategorien. Abbildung 1 soll zur Visualisierung dieser Zusammenhänge beitragen.

Abbildung 1: Kategorien freier und nicht freier Software[11]



[8] Vgl. Linauer, 2002, S. 1.
[9] Siehe Kapitel 2.
[10] Vgl. O´Sullivan et al., S. 38.
[11] http://www.fsf.org/philosophy/categories.html.

Free Software

Bei Free Software handelt es sich um Software, die von jedem verwendet, kopiert, verändert oder unverändert, gratis oder gegen eine Gebühr weitergegeben werden kann. Die Voraussetzung dafür ist der freie Zugang zum Quellcode der Software. Wenn ein Programm frei ist, kann es in freie Betriebssysteme wie GNU/Linux integriert werden. Der Begriff „frei" bezieht sich nicht auf den Preis der Software, sondern auf die Verfügbarkeit des Quellcodes. Im Gegensatz dazu verwenden Hersteller proprietärer Software die Bezeichnung freie Software mit Bezug auf den Programmpreis. Damit ist entweder gemeint, dass User eine Kopie des Binärcodes[12] kostenlos zur Verfügung gestellt bekommen, oder dass eine Kopie am gekauften Computer installiert ist. Beides hat jedenfalls nichts mit freier Software im oben genannten Sinne zu tun[13].

Open Source Software

Die Begriffe Open Source Software und Free Software bezeichnen mehr oder weniger dasselbe und werden individuellen Präferenzen entsprechend verwendet.

Copylefted Software

Copylefted Software ist freie Software, deren Vertriebsbedingungen verhindern sollen, dass die Hersteller von veränderten Versionen diese bei der Weitergabe zusätzlichen Restriktionen unterwerfen. Mit anderen Worten: jede Kopie, die von freier Software angefertigt wurde, muss freie Software bleiben, auch wenn sie modifiziert wurde[14].

Non-Copylefted Free Software

Wenn eine Software frei ist, jedoch nicht unter Copyleft steht, können Kopien oder modifizierte Versionen privatisiert werden. Kommerzielle Softwarehersteller können so unfreie Versionen herstellen und diese als proprietäres Softwareprodukt verkaufen, wie das z. B. beim „X Window System" der Fall ist[15].

Public Domain Software

Public Domain Software verfügt über keinen Copyright-Schutz. Die Gründe dafür können sein, dass die Software gesetzlich nicht geschützt wird oder der Herausgeber auf seine Copyrights verzichtet bzw. diese nicht mehr gelten[16].

Bei einem solchen Quellcode handelt es sich um einen speziellen Fall von nicht unter Copyleft stehender freier Software, d. h. Kopien oder veränderte Versionen sind teilweise nicht frei. Es kann sein, dass sich ein Programm in der Public Domain befindet, der dazugehörige Quellcode aber nicht erhältlich ist und das Programm deswegen nicht als freie Software bezeichnet werden kann. Mittlerweile befindet sich ein Großteil der freien Software nicht mehr in der Public Domain,

[12] Das binäre Zahlensystem repräsentiert eine Abfolge von Nummern, die nur die Ziffern 0 und 1 als Stellen verwenden. Ein typisches Beispiel dafür sind eben Computerprogramme. So beziehen sich kompilierte Applikationen meist einfach auf den Binärcode, der, im Gegensatz zum Quellcode, nicht in Form von Textdateien existiert.

[13] Vgl. http://www.fsf.org/philosophy/categories.html.

[14] Mehr dazu unter Kap. 1.2.1.

[15] Vgl. http://www.fsf.org/philosophy/categories.html.

[16] Vgl. Grassmuck, 2000, S. 88.

sondern steht unter Copyright, und die Hersteller verwenden freie Softwarelizenzmodelle, um Usern den freien Gebrauch zu ermöglichen[17].

„Public domain" ist ein rechtlicher Begriff und meint „kein Copyright" und nicht „frei" oder „kostenlos erhältlich", wie irrtümlicherweise oft angenommen wird. Jeder Text – und auch jedes Programm – wird automatisch unter Copyright gestellt. Wenn nun ein Hersteller will, dass sich sein Programm in der Public Domain befindet, muss er dazu eigene rechtliche Schritte unternehmen, um sein Urheberrecht offiziell abzulehnen; andernfalls steht das Programm unter Copyright[18].

GPL covered Software
Die GNU[19] General Public License (GNU GPL) verfügt über eine Anzahl von Vertriebsbestimmungen, die ein Programm unter Copyleft stellen. Das GNU-Projekt[20] verwendet die GPL als Vertriebsbedingungen für beinah die gesamte GNU-Software[21].

Halbfreie Software
Halbfreie Software ist nicht freie Software, die den Usern aber die Möglichkeit bietet, das Programm für nichtkommerzielle Zwecke kostenlos zu verwenden, zu kopieren, weiterzugeben und zu verändern. Halbfreie Software ist zwar grundsätzlich proprietärer Software vorzuziehen, das Problem bei halbfreier Software liegt aber darin, dass diese nicht in einem freien Betriebssystem verwendet werden kann[22].

Der Sinn der Einschränkungen von Copyleft liegt darin, zu verhindern, dass Usern durch andere, zusätzliche Einschränkungen der freie Gebrauch der Software verweigert wird. Halbfreie Programme verfügen aber über solche zusätzlichen Restriktionen, die ausschließlich zur Verfolgung eigennütziger Ziele entworfen werden.

Die Verwendung von halbfreien Programmen in einem freien Betriebssystem ist nur deswegen nicht möglich, da die Distributionsbestimmungen für das Betriebssystem als Ganzes die Verknüpfung der Distributionsbestimmungen aller in ihm enthaltenen Programme darstellt. Die Beifügung nur eines halbfreien Programms würde daher das ganze Betriebssystem halbfrei machen. Die Gründe für die Vermeidung einer solchen Entwicklung sind folgende: Freie Software sollte auch für kommerzielle Unternehmen zur Verfügung stehen, und um Betriebe zur Verwendung des GNU-Systems zu bewegen, dürfen keine halbfreien Programme darin enthalten sein. Ebenso ist der kommerzielle Vertrieb von freien Betriebssystemen, das GNU/Linux-System eingeschlossen, sehr wichtig, und User schätzen die Vorteile kommerzieller CD-ROM-Distributionen. Die Integration eines einzigen halbfreien Programms in ein Betriebssystem würde eine kommerzielle Distribution unmöglich machen[23].

---

[17] Vgl. http://www.fsf.org/philosophy/categories.html.
[18] Vgl. http://www.fsf.org/philosophy/categories.html.
[19] GNU ist ein Wortspiel und steht für „GNU´s not Unix"; es handelt sich dabei um ein freies Betriebssystem. Mehr über die Entstehung von GNU und das GNU-Projekt befindet sich in Kap. 1.2.1.
[20] Siehe Kapitel 1.2.1.
[21] Vgl. http://www.fsf.org/philosophy/categories.html.
[22] Mehr dazu unter http://www.fsf.org/philosophy/categories.html.
[23] Vgl. http://www.fsf.org/philosophy/categories.html.

Private Software
Bei privater Software handelt es sich um Software, die nur für einen einzigen Anwender, z. B. eine Organisation oder ein Unternehmen, entwickelt worden ist. Dieser verwendet das Programm, ohne das Programm selbst bzw. den Quellcode oder den Binärcode der Öffentlichkeit weiterzugeben. Private Software ist daher in gewissem Sinne auch freie Software, da der Eigentümer den Quellcode und alle Rechte darauf besitzt. Generell wird private Software aber nicht zur freien Software gezählt[24].

Proprietäre Software
Proprietäre Software ist weder freie noch halbfreie Software. Ihre Verwendung unterliegt klaren Richtlinien, und Veränderungen sind verboten oder zumindest stark eingeschränkt. Der Quellcode von proprietären Programmen wird, im Gegensatz zu den Codes freier Software, nicht offen gelegt und unterliegt strengen Lizenzbestimmungen und Copyrights.

Freeware
Freeware ist Software, die zwar unter Copyright steht, aber vom Entwickler oftmals ohne lizenzrechtliche Bestimmungen freigegeben wird. Die Weitergabe von Freeware ist zwar erlaubt, jedoch nicht die Vornahme von Veränderungen, was aufgrund der Veröffentlichung der Software ohne Angabe des Quellcodes ohnehin kaum möglich wäre.

Shareware
Hersteller von Shareware erlauben die Weitergabe von Kopien, fordern die User aber auf, bei regelmäßigem Gebrauch dieser Kopien eine Gebühr dafür zu entrichten. Shareware ist keine freie bzw. halbfreie Software, da meist der Quellcode nicht zugänglich gemacht wird und für freie Software grundsätzlich keine Gebühr zu entrichten ist.

Kommerzielle Software
Bei kommerzieller Software handelt es sich um Software, die zur Vermarktung entwickelt wurde. Kommerziell bedeutet aber nicht gleich proprietär. Obwohl die meiste kommerzielle Software proprietär ist, gibt es auch kommerzielle freie bzw. nichtkommerzielle unfreie Programme.

Ein Beispiel für kommerzielle freie Programme ist GNU Ada. GNU Ada steht unter der GNU General Public License und jede Kopie ist freie Software, während die Entwickler zur gleichen Zeit Supportverträge verkaufen. GNU Ada wird dadurch zur kommerziellen Software und bleibt dennoch freie Software[25].

Unterschied zwischen Open Source und Open Standard

Bei Open Source Software handelt es sich um Programme, bei denen zum einen der Quellcode frei zugänglich ist und darüber hinaus vom User verwendet, weitergegeben, kopiert, modifiziert und in modifizierter Form verteilt werden kann. Es besteht weiters für jeden die Möglichkeit, sich bei Open-Source-Projekten an der Entwicklung, Veränderung und Optimierung von Software zu beteiligen. Das hier entstehende Wissen wird als Allgemeingut betrachtet und frei zur Verfügung

[24] Vgl. http://www.fsf.org/philosophy/categories.html.
[25] Vgl. http://www.fsf.org/philosophy/categories.html.

gestellt. Open Standards hingegen sind öffentlich erhältliche Spezifikationen, um die Kompatibilität zwischen verschiedenen Hardware- und Softwarekomponenten zu verbessern[26]. Jeder, der das notwendige Know-how und die erforderlichen Ressourcen besitzt, hat die Möglichkeit, Lösungen zu entwickeln, die mit denen anderer Erzeuger kompatibel sind[27]. Beispiele für Open Standards bei Hardware sind: ISA (Industry Standard Architecture), PCI (Peripheral Component Interconnect), AGP (Accelerated Graphics Port). Open Standard Software sind beispielsweise HTML (Hyper Text Markup Language), SQL (Structured Query Language), IP (Internet Protocol) oder TCP (Transmission Control Protocol).[28]

1.2 OPEN SOURCE SOFTWARE-LANDSCHAFT UND -STRUKTUREN

1.2.1 Die Geburtsstunde von Open Source Software: das GNU-Projekt

Als sich Anfang der 1980er Jahre ein Entwicklungstrend in Richtung proprietärer Software abzeichnete, wurde 1984 als Antwort auf die zunehmende Kommerzialisierung von Software das GNU-Projekt ins Leben gerufen. Gründer dieses Projekts war Richard Stallman, der damit eine Gegenbewegung zur proprietären Softwareprogrammierung erzeugen wollte.

Ziel des Projekts war es, ein Betriebssystem zu schreiben, das von den Funktionen her äquivalent zu Unix[29] ist, aber keinen geschützten Code enthält und auf Basis freier Kooperationen weiterentwickelt werden kann („GNU“ ist die Bezeichnung für „GNU´s not Unix“). Unix wurde deswegen ausgewählt, „weil es sich bewährt hatte, weil es portabel ist und weil es bereits eine aktive weltweite Unix-Gemeinde gab, die durch seine Kompatibilität leicht zu GNU wechseln konnte“[30].

Um die notwendigen Finanzierungsmöglichkeiten ausreichend ausschöpfen und erschließen zu können, wurde 1985 von Richard Stallman die Free Software Foundation (FSF) gegründet. Die FSF ist zuständig für die Distribution der GNU-Software, wobei die Mittel aus den Verkäufen sowie die Einnahmen aus Geld- und Sachspenden für die Entwicklung von Programmen, die für ein vollständiges Betriebssystem noch notwendig waren, verwendet wurden[31].

Das GNU-Manifest

Das GNU-Manifest wurde von Stallman zu Beginn des Projekts geschrieben und sollte als Aufruf zur Unterstützung und Mitarbeit dienen. Er beschrieb in diesem Manifest seine Beweggründe für die Entwicklung von freier Software: „I consider that the golden rule requires that if I like a program

[26] Kompatibilität hier meint, die Möglichkeit zu haben, ein Programm auf verschiedenen Computern ausführen zu können, ohne das Programm oder den Computer verändern zu müssen.
[27] Vgl. http://www.fact-index.com/o/op/open_standard.html.
[28] Vgl. http://www.fact-index.com/o/op/open_standard.html.
[29] Unix ist die Bezeichnung für das Betriebssystem, das in den 1960er Jahren von AT&T entwickelt wurde. Unix ist auf verschiedenen Plattformen einsetzbar und unterstützt eine interaktive Computernutzung. Im Zusammenhang mit Unix wurde erstmals das Konzept der Source-Code-Kompatibilität eingeführt. Da AT&T zu Beginn als staatlich reguliertes Telefonmonopol Unix nicht kommerziell vermarkten durfte und daher auch kein Support angeboten wurde, bildete sich unter den Usern rasch eine Community heraus.
[30] Grassmuck, 2000, S. 41.
[31] Vgl. Grassmuck, 2000, S. 42.

I must share it with other people who like it."[32] Er verurteilte darin Anbieter von Software, die ihr Wissen schützen und erklärte die Gründe, warum Menschen sich an der Entwicklung von Open Source Software beteiligen und welche Möglichkeiten es für jeden einzelnen gibt, sich an dieser Teamarbeit zu beteiligen. Ebenfalls Bestandteil des Manifests sind die Vorteile, die die Verwendung von freier Software für die User mit sich bringt[33].

Das oberste Ziel des GNU-Projekts war ein vollständiges Betriebssystem. Alle dazu notwendigen Bestandteile wurden systematisch in eine Task List eingetragen und der Reihe nach erarbeitet[34]. Außerdem wurden überall dort, wo es möglich war, Teile von bereits bestehender freier Software angepasst und genutzt. Das GNU-System enthält daher Programme, die nicht GNU-Software sind bzw. Programme, die von anderen Personen stammen und für andere Projekte und Zwecke entwickelt wurden, die aber verwendet werden können, weil es sich dabei um freie Software handelt[35].

Das maßgebliche Instrument zur Absicherung der Software, ist die dazugehörige Lizenz. Die GNU General Public License gewährt den Usern den Zugang zum Quellcode, sie erlaubt die Anfertigung von Kopien und deren Verteilung, sowie die Vornahme von Änderungen und die Weitergabe des veränderten Programms unter denselben Bedingungen[36]. Dadurch, dass die Weitergabe unter denselben Bedingungen zu erfolgen hat, wird verhindert, dass ursprünglich freie Software privatisiert wird. Nicht vorgeschrieben wird jedoch eine unentgeltliche Verteilung der Software[37].

Im Jahre 1990 war das GNU-System vollständig, die einzige noch fehlende Komponente war ein Kernel[38]. Der 1991 von Linus Torvalds entwickelte Linux-Kernel wurde schließlich in die GNU-Umgebung eingefügt und seitdem gibt es das vollständige und vor allem freie Betriebssystem GNU/Linux[39].

Um zu verhindern, dass GNU-Software zu proprietärer Software gemacht wird, wurde die Methode des „copyleft" eingeführt. Copyleft nutzt das Urheberrechtsgesetz, wendet es aber so an, dass es nicht mehr ein Instrument zur Privatisierung von Software ist, sondern ein Mittel, um Software frei zu erhalten.

Die zentrale Idee von Copyleft ist, dass jeder das Programm verwenden, kopieren und verändern und die veränderte Software auch vertreiben darf. Was der User aber nicht darf, ist eigene Einschränkungen hinzuzufügen. Für die Effektivität von Copyleft ist es wichtig, dass auch modifizierte Versionen frei erhältlich sind.

---

[32] Stallman, 1985: http://www.gnu.org/gnu/manifesto.html
[33] Mehr dazu unter: http://www.gnu.org/gnu/manifesto.html.
[34] Vgl. Grassmuck, 2000, S 43f.
[35] Vgl. Stallman: http://www.gnu.org./gnu/thegnuproject.html.
[36] Mehr dazu in Kapitel 4.1.
[37] Vgl. Grassmuck, 2000, S 44.
[38] Ein Kernel ist der kritische Mittelpunkt eines Betriebssystems und kontrolliert z. B. den Gebrauch des Prozessors oder Hardware-Funktionen und steuert die Kommunikation zwischen den verschiedenen Programmen innerhalb eines Betriebssystems.
[39] Vgl. Grassmuck, 2000, S 44.

Die konkrete Umsetzung von Copyleft erfolgt für fast die gesamte GNU-Software über die GNU General Public License (GNU GPL)[40]. Auch GNU-Handbücher verfügen über ein Copyleft, welches jedoch weniger komplex ist als die GNU GPL[41].

Linux

Linux ist ein freies Betriebssystem, das Unix sehr ähnlich ist. Linux wurde 1991 an der Universität von Helsinki entwickelt und enthält alle wichtigen Merkmale einer Open Source Software. Es steht unter der GNU GPL, wodurch der freie Zugang zum Quellcode für jeden User möglich ist.

Linux ist eigentlich nur der Kernel des Systems und ist im Wesentlichen für die Daten- und Speicherverwaltung zuständig. Andere wichtige Komponenten stammen aus verschiedenen anderen GNU-Projekten. Das Gesamtergebnis daraus ist die vollständige Betriebssystemumgebung GNU/Linux[42].

Da alle Anwendungen zwar vom Kernel kontrolliert werden, im Grunde aber eigenständige Programme sind, „stürzt“ bei Fehlern nicht der gesamte Rechner ab, sondern nur die betroffene Funktion[43].

Linux war ursprünglich der Name des Kernels, der von Linus Torvalds entwickelt worden war. Kernel sind bis zu einem gewissen Grad austauschbar, jedoch laufen die meisten Open Source Software-Applikationen auf Basis eines Linux-Kernels[44].

Aufgrund der Modularität von GUN/Linux lassen sich je nach Bedarf verschiedene Komponenten zu einem Betriebssystem zusammenfassen. Einige Unternehmen haben sich darauf spezialisiert, fertige und auf verschiedenen Einsatzszenarien optimierte GNU-Programmpakete zusammenzustellen – diese werden als Distributionen bezeichnet.

Eine Distribution von Linux umfasst neben dem Linux-Kernel alle Open Source Software-Komponenten, die notwendig sind, um eine vollständige Funktionalität des Betriebssystems herstellen zu können. Es gibt verschiedene Distributoren, wobei jeweils nur der Linux-Kernel gleich bleibt, die jeweiligen Komponenten und Konfigurationen jedoch variabel sind[45].

Der Linux-Kernel kann im Internet frei unter der Adresse http://www.kernel.org heruntergeladen werden, die weiteren Applikationen von den jeweiligen Websites der Communities.

[40] Mehr dazu in Kapitel 1.3.1.
[41] Vgl. Stallman: http://www.gnu.org/gnu/thegnuproject.html.
[42] Vgl. Linauer, 2002. S. 2.
[43] Vgl. ebenda, S. 2 f.
[44] Vgl. Sayo, Wong, S. 27.
[45] Vgl. Sayo, Wong, S. 27.

Vorteile von Linux

Stabilität
Der Quellcode bzw. der Code des Linux-Kernels ist frei zugänglich und wird durch zahlreiche User in aller Welt entwickelt und verbessert (weltweite Community). Die Programme werden schon sehr früh, in einer „unfertigen" Version, freigegeben. Einerseits können dadurch rechtzeitig Korrekturen vorgenommen werden und anderseits das Programm um neue, zusätzliche Funktionen ergänzt werden. Darüber hinaus ist die Wahrscheinlichkeit, einen Fehler zu entdecken, bei einer größeren Anzahl von Programmierern höher. Im Endeffekt ist freier Code somit meist mit weniger Fehlern behaftet als traditionell entwickelte Software.

Sicherheit
Derzeit existieren für Linux noch kaum gefährliche Viren. Die Gründe dafür liegen einerseits darin, dass Linux noch nicht so weit verbreitet ist, aber auch im schnellen Umgang mit Sicherheitslücken und in der Tatsache, dass Linux-Benutzer nur Zugang auf ihre privaten Dateien und nicht auf Systemdateien haben. Da der Quellcode laufend von vielen Personen überprüft wird, werden Programmierfehler, die ein Sicherheitsrisiko darstellen, in der Regel sehr schnell entdeckt[46].

Kosten
Linux-Distributionen sind meist kostenlos und über das Internet downloadbar bzw. sind sie sehr kostengünstig zu beziehen[47].

Nachteile von Linux

Ein wesentlicher Nachteil von Linux ist die mangelnde Hardwareunterstützung. Dies ist in erster Linie darauf zurückzuführen, dass Hardwarehersteller bisher noch wenig Interesse an Linux-Treibern[48] gezeigt haben. In letzter Zeit ist jedoch eine positive Entwicklung erkennbar, da die Verbreitung von Linux steigt und die Industrie auf dieses wachsende Kundensegment nicht verzichten kann[49].

1.2.2 Entstehung von Open Source Software

Open Source Software entsteht durch den „formlosen" Zusammenschluss einer Gruppe von Entwicklern. Teilnehmen an einem Open-Source-Projekt kann prinzipiell jeder, der Interesse daran hat und die notwendigen Fähigkeiten dafür besitzt. Es gibt keine monetäre Entlohnung für die Mitarbeit am Projekt und es existieren auch keine Geheimhaltungsvorschriften hinsichtlich der Ergebnisse, da das hier gesammelte Wissen allen gehört und die Urheberrechte von den Entwicklern abgetreten werden[50].

---

[46] Vgl. http://www.nimm-linux.tk.
[47] Vgl. http://www.nimm-linux.tk.
[48] Unter einem Treiber versteht man ein spezielles Softwaremodul, das das Ansteuern der Hardware eines Herstellers durch das Betriebssystem erlaubt. Betriebssystemhersteller integrieren zwar die gängigste Hardwareunterstützung, diese kann jedoch auch nur eine Momentaufnahme darstellen und neue Hardware nicht berücksichtigen.
[49] Vgl. http://www.nimm-linux.tk.
[50] Vgl. Grassmuck, 2000, S. 51.

Der Quellcode ist somit für jeden einsehbar und veränderbar. Im Gegensatz dazu wird bei proprietärer Software der Quellcode nicht offen gelegt und auch Veränderungen durch den User werden technisch unmöglich gemacht und sind aus lizenzrechtlichen Gründen auch nicht erlaubt.

Daraus ergibt sich für freie Software ein wesentlicher Vorteil. Durch die Offenlegung des Quellcodes werden Programme ständig erweitert, abgewandelt und oftmals verbessert. Es entstehen dadurch neue Anwendungen, die vielfach einsetzbar sind. Für die hohe Qualität der Software sorgt auch die vorzeitige Prüfung und Verbesserung der Programme, die bereits in einer frühen Entwicklungsphase den Usern zugänglich gemacht werden. Das Ziel hierbei ist, relativ rasch über eine stabile und fehlerfreie Software verfügen zu können: die Wahrscheinlichkeit, Schwachstellen und Fehler zu entdecken ist, umso größer, je mehr Personen bzw. Experten den Quellcode durchsehen und überprüfen[51].

Die Entwicklung von Open Source Software wird vom Gedanken einer gemeinsamen Problemlösung getragen: mit der frühzeitigen Veröffentlichung eines Projekts wird angestrebt, eine Vielzahl von Personen zu motivieren, an der Lösung praktischer Probleme – und sei es nur in Teilbereichen einer Software – mitzuarbeiten[52].

1.2.3 Ablauf von Open-Source-Projekten

Ein Open-Source-Projekt entsteht meist nicht in Auftrag, sondern wird zur Lösung eines in der Arbeitspraxis aufgetauchten – und mit proprietärer Software nicht zufriedenstellend abgedeckten – Problems gestartet. Durch die Bekanntgabe des Projekts bzw. des Problems in einem sehr frühen Stadium entsteht um den Initiator meist bald ein Projektteam bestehend aus Personen, die ebenfalls an der Lösung des Problems interessiert sind und auch bereit sind, daran mitzuarbeiten. Vergrößert sich im Verlauf des Projekts die Zahl der Mitarbeiter, übernehmen einzelne Personen als „Maintainer“ die Koordination und Verantwortung für Teile des Projekts. Ebenso bildet das ursprüngliche Projektteam das „Core-Team“, wenn sich Rahmen und Umfang des Projekts ausweiten. Das Core-Team setzt sich üblicherweise aus Personen zusammen, die sehr stark in das Projekt involviert sind und sich aktiv an der Umsetzung beteiligen. Das Core-Team entscheidet über den weiteren Verlauf des Projekts, Design-Fragen und die Bearbeitung spezieller Problembereiche[53].

Bei größeren Projekten erfolgt eine Gliederung in funktionale Einheiten, wobei die Verantwortung für diese Einheiten auf einen oder mehrere Maintainer übertragen wird. An der Mitarbeit an einem solchen Teilprojekt sind bis zu hundert Entwickler beteiligt. Erfolgen Änderungen in den Einzelprojekten, werden diese an das Core-Team weitergeleitet und in den „Source-Baum“[54] integriert[55].

[51] Vgl. Linauer, 2002, S. 2.
[52] Vgl. Grassmuck, 2000, S. 53.
[53] Vgl. Grassmuck, 2000, S. 53 f.
[54] Der Source-Baum enthält alle Informationen über die Programme, daher werden Änderungen an der Software bzw. Daten für Releases in diesen integriert.
[55] Vgl. Grassmuck, 2000, S. 54.

Die Personen, die rund um das Core-Team weltweit an der Entwicklung und Verbesserung der Software beteiligt sind, werden als „Community" bezeichnet. Die Mitglieder der Community leisten ihren Beitrag durch ein- oder mehrmalige Mitarbeit an verschiedenen Projekten. Die Community bildet gewissermaßen die Basis eines Open-Source-Projekts, sodass in letzter Zeit auf die Pflege der Community verstärkt Wert gelegt wurde (z. B. wird auf das Recht auf Namensnennung stärker geachtet). Häufig herrscht in freien Projekten eine flexible Arbeitsaufteilung, wobei jeder Beteiligte seine Stärken und Interessenschwerpunkte in den Vordergrund stellen kann[56].

Die Entscheidungsfindung erfolgt nach dem Motto „rough consensus and running code"[57], d. h. eine Entscheidungsfindung auf kooperativer, gemeinschaftlicher Basis im Sinne der Entwicklung eines optimal einsetzbaren Codes steht im Vordergrund. Die Mitglieder des Core-Teams übernehmen dabei nicht die Funktion von Vorgesetzten, und Entscheidungen werden durch offene Diskussion und Mehrheitsbeschluss getroffen, wobei es natürlich auch zu Konflikten und in seltenen Fällen zur Aufspaltung des Projekts kommt[58].

Die wesentlichen Kommunikationsmittel für das weltumspannende Kooperationsnetzwerk sind Mailinglisten und Newsgroups. Informationen über die Projekte und die verfügbaren Ressourcen finden sich auf Websites.

Das Concurrent Versions System (CVS) ist das wichtigste Tool zur kooperativen Quellcode-Verwaltung. CVS wird bei freier Software standardmäßig eingesetzt und ist notwendig für eine effiziente Revisionsverwaltung: berechtigte Personen können eine Koordinationsfunktion ausüben, wenn mehrere Personen an den selben Dateien arbeiten und sie können sich einen Überblick über vorgenommene Modifikationen verschaffen. Generell kann jeder Einblick in die Source-Bäume nehmen und sich die aktuellste Version kopieren[59].

Wenn eine Person selbst an der Fortschreibung des Codes interessiert ist, muss sie sich registrieren lassen, damit sie für andere Entwickler erreichbar ist. Danach kann mit der Bearbeitung begonnen werden, indem eine aktuelle Fassung der entsprechenden Dateien eines gewünschten Moduls sodann vom CVS aus dem Repository[60] in einen eigenen Verzeichnisbaum kopiert wird. Danach können die Dateien vom Entwickler beliebig verändert werden. Entsprechen die Änderungen seinen Vorstellungen, schickt er die betreffenden Dateien in das Repository zurück, wodurch andere Entwickler Zugang zu den Veränderungen erhalten[61].

Aufgrund der Offenheit gegenüber Fehlern, die in solchen Projekten herrscht, besitzt eine solcherart entwickelte Software einen sehr hohen Grad an Fehlerfreiheit. Anders als bei kommerziellen Softwareherstellern, die unter dem Diktat der Wirtschaftlichkeit arbeiten und den Entwicklungsaufwand möglichst gering halten müssen, wird ein entdeckter Fehler allen bekannt gemacht und die Community zur gemeinsamen Problemlösung aufgefordert[62].

[56] Vgl. ebenda S. 55.
[57] Clark, 1992 (zit. nach: Grassmuck, 2000, S. 55).
[58] Vgl. Grassmuck, 2000, S. 55 f.
[59] Vgl. ebenda, S. 57.
[60] Ein Repository ist ein gemeinsames Verzeichnis – meist eine Datenbank – in dem sich die gesammelten Dateien befinden.
[61] Vgl. Grassmuck, 2000, S. 57.
[62] Vgl. Grassmuck, 2000, S. 58.

Eine Release[63] von Open Source Software erfolgt frühzeitig und in kurzen Abständen. Da die Entwickler nicht an Aufträge oder Termine gebunden sind, können die Releases zwanglos untereinander besprochen und optimiert werden, bis sie die erforderliche Stabilität aufweisen. Bevor jedoch eine neue Version freigegeben wird, wird zunächst der Source-Baum, in dem alle vorgenommenen Veränderungen enthalten sind, für Änderungen gesperrt. Die Modifikationen werden danach getestet, bei Bedarf korrigiert und auf ihre Kompatibilität mit den bestehenden Bestandteilen überprüft. Erst wenn diese Tätigkeiten abgeschlossen sind, wird die Release offiziell freigegeben[64].

Bei Open-Source-Projekten handelt es sich meist um lose Arbeitsgemeinschaften ohne eine nach außen gültige Rechtsform. Die Entstehung einer Rechtsform erweist sich dann als erforderlich und vorteilhaft, wenn Kooperationen mit anderen Unternehmen eingegangen werden, Beteiligungen entstehen oder die Vertretung der eigenen Interessen vor Gericht notwendig wird. Beispiele dafür sind die Apache Software Foundation (ASF), die Free Software Foundation (FSF) oder Software in the Public Interest (SPI)[65].

1.2.4 Akteure im Bereich Open Source Software

Free Software Foundation

Die Free Software Foundation (FSF) fördert das Recht der User, Computerprogramme zu verwenden, studieren, kopieren, verändern und weiterzugeben. Sie wurde 1985 gegründet, Gründer und Präsident ist Richard Stallman. Die FSF engagiert sich weiters im Bereich der Entwicklung und Verwendung von freier Software, besonderer Schwerpunkt liegt hier beim GNU/Linux-Betriebssystem, und freier Dokumentation[66].

Im Gegensatz zu anderen Organisationen vertreibt FSF nicht nur bereits vorhandene Programme, sondern legt den Fokus auf die Entwicklung von neuer freier Software.

Neben der Entwicklung von Software ist die Free Software Foundation bestrebt, freie Software zu schützen und zu unterstützen. Sie verkauft Kopien von GNU-Software und Handbüchern und akzeptiert Spenden zur Unterstützung der Weiterentwicklung von GNU. Die Haupteinnahmen der FSF stammen aus diesem Vertriebsservice[67].

Open-Source-Initiative

Die Open-Source-Initiative (OSI) wurde im Februar 1998 von Eric Raymond und Bruce Perens gegründet, weitere Initiatoren waren Hall und Augustin, Präsident ist Eric Raymond. Die OSI ist ein Non-Profit-Unternehmen, das durch ein umfassendes Zertifizierungsprogramm die Open-Source-Bewegung im Sinne der Community zu steuern und fördern versucht. Interessenten erhalten über OSI Informationen über zertifizierte Softwareprodukte, das OSI-Zertifizierungsprogramm und Einsicht in gängige Lizenzmodelle[68].

---

[63] Bei einer Software Release wird eine neue oder verbesserte Version des Systems oder Programms geschaffen und den Usern zur Verfügung gestellt. Die Art der Distribution wird dabei individuell von den Herstellern festgelegt.
[64] Vgl. Grassmuck, 2000, S. 60.
[65] Vgl. ebenda, S. 61 f.
[66] Vgl. http://www.fsf.org/fsf/fsf.html.
[67] Vgl. http://www.fsf.org/fsf/fsf.html.
[68] Vgl. http://www.opensource.org/index.php.

Damit sich User sicher sein können, dass eine bestimmte Software auch tatsächlich Open Source Software ist, vergibt OSI das Zertifikat „OSI Certified“ an Programme, die unter einer Lizenz laufen, die mit der Open-Source-Definition konform sind[69].

Eine Liste der Lizenzen, die von OSI anerkannt wurden, ist erhältlich unter http://www.opensource.org/licenses/index.html.

Free Software Consortium (FSC)

Das FSC wurde von einer Gruppe von Anhängern und Verfechtern der Open Source Software-Idee gegründet, um so den Anforderungen des Marktes durch den Gebrauch von freier Software in kommerziellen Projekten begegnen zu können. Macht, Wissen und Ressourcen sind dezentralisiert und jedes Mitglied hat eine klar definierte und dennoch dynamische Rolle innerhalb des Konsortiums. Die Organisationsstrukturen werden von FSC folgendermaßen dargestellt: „The Consortium’s organizational structures are like organs and limbs of a living organism. They can handle complex tasks for general or specific purposes. But, unlike most living organisms, the organs and limbs of The Free Software Consortium may be removed or entirely new ones created to satisfy current needs.“[70] Das Free Software Consortium sieht demnach seine Organisation als lebenden Organismus und bezeichnet seine Mitglieder quasi als die Organe und Gliedmaßen dieses Organismus, die mit komplexen Aufgaben für verschiedene Zwecke beschäftigt sind. Der Unterschied zu einem lebenden Organismus besteht jedoch darin, dass die Organe und Gliedmaßen, sprich: die Mitglieder, jederzeit ausgetauscht und neu zusammengesetzt werden können.

Die Grundsätze der FSC heißen Freiheit und Recht: im Gegensatz zu anderen Gesellschaften wird die Arbeitsweise nicht vorgegeben. Die Mitglieder werden ermutigt, ihre eigenen Methoden zur Erzielung von Ergebnissen einzusetzen, solange sie dabei nicht die Grundsätze und Richtlinien des Konsortiums verletzen[71].

SourceForge.net

SourceForge.net ist die weltweit größte Website für die Entwicklung von Open Source Software und bietet freie Plattform für zehntausende Projekte. Die Mission von SourceForge.net ist die Stärkung der Open-Source-Community durch die Bereitstellung einer zentralen Anlaufstelle für Open-Source-Entwickler, die dadurch die Entwicklung von Open Source Software besser kontrollieren und steuern können. Um diese Mission erfüllen zu können, wird von SourceForge.net eine Vielzahl von Serviceleistungen speziell für betreute Projekte und für die Open-Source-Community im Allgemeinen angeboten[72].

Der Eigentümer von SourceForge.net ist OSTG, Inc. (Open-Source-Technology-Group), lt. Eigendefinition „the most dynamic community-driven media network on the Web“[73].

---

[69] Vgl. http://www.opensource.org/docs/certification_mark.php.
[70] http://www.fsc.cc/taxonomy_menu/1/1.
[71] Vgl. http://www.fsc.cc/taxonomy_menu/1/1.
[72] Vgl. http://www.sourceforge.net/docman/display_doc.php.
[73] http://www.sourceforge.net/docman/display_doc.php, mehr dazu unter www.ostg.com/.

1.3 LIZENZMODELLE

Bei der Installation jeder Software erscheint der dazugehörige Lizenztext. Handelt es sich dabei um proprietäre Software, wird der User darin informiert, dass er nur ein eingeschränktes Nutzungsrecht besitzt, dieses Programm nur auf einem Rechner installieren darf u. v. m.

Open Source Software unterscheidet sich dadurch, dass das Copyright der Autoren beansprucht wird, um gleichzeitig spezifische Nutzungsfreiheiten in den Lizenzen festzulegen. Die genauen Einzelheiten sind variabel, drei zentrale Punkte sind aber in jedem Lizenzmodell enthalten. Diese beziehen sich zum einen auf die Beifügung des Quellcodes zum Binärcode[74] des Programms und zum anderen auf das Recht, die Software zu kopieren und weiterzugeben und auf das Recht, die Original-Software zu verändern und die veränderte Version zu verbreiten[75].

Der wesentliche Unterschied besteht demnach darin, dass bei proprietärer Software zwar der Gebrauch erlaubt ist, im Gegensatz zu freier Software aber nicht die Erzeugung von Kopien und modifizierten Versionen.

Damit freie Software auch wirklich frei zugänglich bleibt, ist der Einsatz von freien Softwarelizenzen notwendig. Dasselbe gilt für den Erhalt von freier Dokumentation: Der Leser soll die Möglichkeit haben, veröffentlichte Dokumentationen zu kopieren und weiterzugeben. Um dies garantieren zu können, muss jede Art von Dokumentation – ebenso wie Software – unter Copyleft-Lizenzen gestellt werden.

Das Ziel von freien bzw. von Open-Source-Lizenzen ist demnach, den freien Zugang zum Quellcode sowie die Freiheit, die Software zu verändern, kopieren und weiterzugeben, zu gewährleisten.

Die bewährtesten und am meisten verbreiteten Lizenzmodelle sind GNU General Public License (GNU GPL) und die BSD-Lizenzen[76]. Daneben gibt es zahlreiche Varianten und eine Vielzahl weiterer offener Lizenzmodelle. Unterschiede zwischen den verschiedenen freien Lizenzmodellen bestehen vor allem bei der Weiterverwendung von Quellcode. Auf der einen Seite gibt es die von der Free Software Foundation idealisierte Copyleft-Lizenz, die GNU GPL-kompatibel ist und den Entwicklern erlaubt, ihre Veränderungen weiterzugeben. Auf der anderen Seite stehen die BSD-Lizenzen, bei denen Veränderungen am Quellcode privatisiert werden können[77].

1.3.1 GNU General Public License (GNU GPL)

Die GNU GPL wird für die meisten GNU-Programme verwendet und für mehr als die Hälfte aller Open Source Software-Pakete[78].

[74] Def. Binärcode: siehe Kap. 1.1.
[75] Vgl. Grassmuck, 2000, S. 85 ff.
[76] BSD steht für „Berkeley System Distribution“, Details unter Pkt. 1.3.4.
[77] Vgl. O´Sullivan, 2003, S. 17.
[78] Vgl. http://www.gnu.org/licenses/lisenses_html.

Ziel dieser Lizenz ist es, die Freiheit der User, die Software zu nutzen und zu verändern, zu erhalten. In der Präambel der GNU GPL wird der Zweck der Lizenz folgendermaßen beschrieben: „The GNU General Public License is intended to guarantee your freedom to share and change free software – to make sure the software is free for all its users. [...] Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.“[79]

Stallmans Ziel war nicht, Software einfach zu verschenken, sondern freie Software in einem gewissen Umfang aufzubauen und zu erhalten. Die Programme, die von einer unter GNU GPL stehenden Software abgeleitet werden, müssen ebenfalls unter GNU GPL stehen[80]. Die Intention einer solchen Bestimmung liegt darin, die Kommerzialisierung von freiem Wissen zu verhindern und die Zahl bestehender Open Source Software kontinuierlich zu erhöhen.

GNU GPL wurde von der Free Software Foundation verfasst, um ein rechtsgültiges Modell zur Entwicklung von freier Software als Alternative neben proprietärer Software zu schaffen[81].

GNU GPL ist das am häufigsten verwendete Lizenzmodell für Open Source Software und steht auch bei Open-Source-Projekten an erster Stelle, gefolgt von GNU Lesser General Public License und BSD-Lizenzen[82].

GNU GPL ist auch das von der FSF (Free Software Foundation) am häufigsten verwendete Lizenzmodell, da es die Anforderungen an eine freie Softwarelizenz erfüllt: GNU GPL ist auf der einen Seite eine Copyleft-Lizenz, d. h. der Quellcode von veröffentlichten und modifizierten Programmen muss zugänglich gemacht werden. Auf der anderen Seite ist GNU GPL eine auto-kompatible Lizenz, was bedeutet, dass Programme, die unter der GPL stehen, auch unter anderen, ähnlichen Lizenzprogrammen verwendet werden können[83].

1.3.2 GNU Lesser General Public License (GNU LGPL)

Die GNU LGPL, ursprünglich GNU Library GPL, wird von einigen GNU-Bibliotheken[84] verwendet und stellt eine spezielle Art von Copyleft dar, die es möglich macht, proprietäre Software mit der GNU-Bibliothek zu verlinken[85]. Die LGPL enthält alle Freiheiten der GPL, d. h. die Bibliothek muss allgemein zugänglich, frei zu kopieren, verändern und weiterzugeben sein. Der Unterschied zur GPL ergibt sich daraus, dass Programme, die diese Bibliothek unter der LGPL aufnehmen, selbst

[79] The GNU General Public License, Version 2, Juni 1991: http://www.opensource.org/licenses/gpl-license.php.
[80] The GNU General Public License, Version 2, Juni 1991: http://www.opensource.org/licenses/gpl-license.php.
[81] Vgl. O´Sullivan et al., S. 42.
[82] Vgl. O´Sullivan, 2003, S. 18.
[83] Vgl. ebenda, S. 15.
[84] Eine Bibliothek ist eine Sammlung von Subprogrammen, die zur Entwicklung von Software verwendet werden. Bibliotheken unterscheiden sich von ausführenden Programmen insofern, als sie keine unabhängigen Programme sind, sondern eher als „Hilfs“-Code anderen unabhängigen Programmen Serviceleistungen anbieten.
[85] Vgl. Stallman: http://www.gnu.org/thegnuproject.html.

nicht den oben genannten Freiheiten unterstehen müssen, mit anderen Worten nicht freie, proprietäre Software sein können[86].

Es ist auch möglich, die originale GNU GPL für Bibliotheken zu verwenden, allerdings ist diese dann nur für freie Software verfügbar. Bibliotheken, die unter der GPL stehen, können demnach nur in Programme integriert werden, die den Kriterien einer Open Source Software entsprechen.

Welche Lizenz nun für eine bestehende Bibliothek verwendet werden soll, ist eine rein strategische Entscheidung und hängt von der jeweiligen Situation ab. Wird die gewöhnliche GPL für eine Bibliothek verwendet, entsteht daraus ein Vorteil für die Entwickler freier Software gegenüber proprietären Entwicklern, da die Bibliothek nur für freie Software verfügbar ist[87].

Der Einsatz der gewöhnlichen GPL ist jedoch nicht bei jeder Bibliothek vorteilhaft. Wenn z. B. die Fähigkeiten und Attribute einer freien Bibliothek proprietärer Software bereits durch andere, alternative Bibliotheken zur Verfügung stehen, ist die Verwendung der LGPL zu empfehlen[88].

1.3.3 GNU Free Documentation License (GNU FDL)

Die GNU FDL ist eine Form von Copyleft, deren Zweck darin besteht, den freien Gebrauch eines Handbuchs, Textbuchs oder anderen Dokuments sicherzustellen. Der User soll weiters die Freiheit haben, dieses Dokument zu kopieren und dies verändert oder unverändert, entgeltlich oder unentgeltlich, weiterzugeben[89].

Ausschlaggebend für die Entwicklung der Free Documentation License war der Gedanke, dass Personen mit dem Verfassen von freier Dokumentation Geld verdienen könnten. Die FDL soll dazu dienen, kommerzielle Herausgeber für die Investition in freie Dokumentation zu gewinnen, ohne dafür auf grundlegende Freiheiten verzichten zu müssen. Die FDL enthält aus diesem Grund verschiedene Aspekte im Zusammenhang mit Cover, Überschrift usw., die sie für solche Herausgeber ansprechend machen soll. Da bereits Interesse seitens kommerzieller Herausgeber von Softwarehandbüchern gezeigt wurde, scheint die FDL gute Chancen zu haben, ein System zu schaffen, in dem kommerzielle Herausgeber Autoren dafür zahlen, dass sie kommerzielle freie Handbücher für freie Software schreiben[90].

1.3.4 BSD-Lizenzen (Berkeley System Distribution)

BSD-Lizenzen wurden ursprünglich für Software verwendet, die von Universitäten, im Speziellen von der Universität von Berkeley, entwickelt wurde.

[86] Vgl. Grassmuck, 2000, S. 97.
[87] Vgl. Stallman, 1999: http://fsf.org/licenses/why-not-lgpl.html.
[88] Vgl. Stallman, 1999: http://fsf.org/licenses/why-not-lgpl.html.
[89] Vgl. Stallman: http://www.gnu.org/licenses/licenses.html.
[90] Vgl. Stallmann, 2000: http://www.gnu.org/licenses/why-gfdl.html.

BSD-Lizenzen erlauben den Usern, die Software beliebig zu verwenden, solange der Original-Lizenzgeber durch einen Original-Copyright-Vermerk im Quellcode angeführt wird und nicht versucht wird, den Original-Lizenzgeber zu verklagen oder für Schäden verantwortlich zu machen[91].

BSD-Lizenzen ermöglichen Veränderungen im Quellcode und eine beliebige Verwendung der veränderten Version, jedoch nur unter Einhaltung der oben angeführten Einschränkungen. Code, der unter einer BSD-Lizenz steht, kann sehr einfach in kommerzielle Applikationen integriert werden, was auch in der Vergangenheit von Herstellern öfters genutzt wurde[92].

Der wesentliche Unterschied zwischen BSD-Lizenzen und der GNU GPL besteht darin, dass bei BSD-Lizenzmodellen keine Verpflichtung besteht, den Quellcode weiterzugeben. Firmen können Veränderungen, die sie am Original-Code vorgenommen haben, geheim halten und müssen ihren Usern auch nicht das Recht gewähren, die Veränderungen einzusehen, weiterzuverändern oder weiterzugeben[93].

BSD-Lizenzmodelle werden in Open-Source-Projekten wie Apache Web Server, BIND, Sendmail oder BSD Unixes verwendet[94].

1.3.5 Entscheidungskriterien für den Einsatz verschiedener Lizenzmodelle

BSD-Lizenzmodell

BSD-Lizenzen sollten verwendet werden, wenn beabsichtigt wird, den Quellcode zur Entwicklung eigener, proprietärer Programme zu verwenden. Diese Programme können danach unter eine beliebige andere – restriktivere – Lizenz gestellt werden, um den veränderten Quellcode geheim halten zu können und einen kommerziellen Vertrieb zu ermöglichen. Beispiele für den Einsatz von BSD-Lizenzen bei Open-Source-Projekten sind Apache, ein sehr erfolgreicher Webserver, oder Apple, der Mac OS X und dessen Basis Darwin unter ein BSD-Lizenzmodell gestellt hat.

GNU GPL

Wird eine Software unter die GPL gestellt, müssen andere Projekte, die Quellcode eines solchen GPL-Projekts aufnehmen, ihr Projekt ebenfalls unter die GPL stellen, was streng genommen eine Einschränkung der freien Lizenzwahl darstellt. Firmen, die ihren Quellcode unter Verschluss halten wollen, können diesen im Falle einer GPL-Lizenzierung auch nicht für ihre eigenen Projekte verwenden und tragen so auch nicht zur Weiterentwicklung des Codes bei. Der Einsatz einer GNU GPL ist daher nur dann empfehlenswert, wenn nicht in weiterer Folge der Vertrieb von modifizierter Software als proprietäres Programm angestrebt wird.

Duale Lizenzierung

Ein Softwareprojekt kann gleichzeitig unter zwei verschiedenen Lizenzen stehen, was sowohl negative als auch positive Effekte mit sich bringt[95].

---

[91] Vgl. Wong, Sayo, S. 30 f.
[92] Vgl. ebenda, S. 31.
[93] Vgl. ebenda.
[94] Vgl. O´Sullivan, 2003, S. 17.
[95] Mehr dazu: O´Sullivan et al, S. 46.

Eine Doppellizenzierung kann dann sinnvoll sein, wenn eine Software für mehrere freie Projekte von Interesse ist und diese durch ein duales Lizenzmodell für viele verschiedene Interessenten zugänglich wird. Würde so ein Programm z. B. nur unter der GPL stehen, wäre es für Projekte mit anderen Lizenzmodellen nicht mehr verwendbar. Eine duale Lizenzierung ist auch dann vorteilhaft, wenn ein Softwarehersteller die modifizierte Version seines Programms, das er unter die GPL gestellt hat, proprietär vertreiben möchte. Da er die freie Lizenzierung nicht rückgängig machen kann (grundsätzlich gilt: einmal GPL, immer GPL!), bleibt nur die Möglichkeit einer Doppellizenzierung. Der Hersteller einer Software ist berechtigt, diese unterschiedlich zu lizenzieren und verstößt nicht gegen lizenzrechtliche Bestimmungen, wenn er seine Software frei und proprietär vertreibt[96].

Klassifizierung von Lizenzen nach FSF

Eine freie Lizenz besitzt lt. FSF folgende Eigenschaften: sie muss die Merkmale einer freien Softwarelizenz aufweisen, es muss sich um eine Copyleft-Lizenz handeln und sie muss kompatibel mit GNU GPL sein. Aufgrund dieser Definition kennt FSF folgende Klassen von Lizenzen[97]:

- **GPL-kompatible, freie Softwarelizenzen**
  (z. B. GNU GPL, GNU LGPL, X 11 License, Public Domain, Modified BSD-License)

- **GPL-inkompatible, freie Softwarelizenzen**
  (z. B. Original BSD-License, OpenSSL License, Academic Free License, Version 1.1, Open Software License, Version 1.0, Apache License, Mozilla Public License)

- **Unfreie Softwarelizenzen**
  Unfreie Softwarelizenzen sind automatisch inkompatibel mit GNU GPL. Es existieren zahlreiche unfreie Softwarelizenzen, als Beispiele werden hier solche angeführt, die oft mit freien Softwarelizenzen verwechselt werden: z. B. Open-Public-License, Apple Public Source License, Version 1.x, AT&T Public License, Microsoft´s Shared Source License etc.

- **Freie Dokumentationslizenzen**
  (z. B. GNU FDL, Free BSD Documentation License, Apple´s Common Documentation License, Version 1.0, Open-Publication-License, Version 1.0)

- **Unfreie Dokumentationslizenzen**
  (z. B. Open-Content-License, Version 1.0, Open-Directory-License)

- **Lizenzen für Arbeiten neben Software und Dokumentation**
  (z. B. GNU GPL, GNU FDL, Design Science License, Free Art License)

[96] Vgl. Jaeger: http://www.linux-magazin.de/Artikel/ausgabe/2001/01/recht/recht.html.
[97] http://www.fsf.org/licenses/license-list.html.

1.4 OPEN-SOURCE-GESCHÄFTSMODELLE

Das folgende Kapitel beschäftigt sich mit Unternehmen, bei denen die Grundlage ihres Geschäftsmodells Open Source Software ist. Die verschiedenen Geschäftsmodelle werden kurz dargestellt, wobei man grundsätzlich zwischen Produkt-, Dienstleistungs- und Mediations-Geschäftsmodellen unterscheidet. Es wird dabei grob umrissen, welche Möglichkeiten es gibt, mit Open Source Software Einnahmen zu erzielen bzw. profitabel zu arbeiten, welche Zielgruppen die verschiedenen Geschäftsmodelle ansprechen und wie Vertrieb und Vermarktung der Produkte erfolgen.

1.4.1 Produkt-Geschäftsmodelle

Bei Produkt-Geschäftsmodellen steht die Entwicklung eines Softwareprodukts, dessen Freigabe unter einer Open Source Software-Lizenz und der Vertrieb und die Vermarktung des Produkts im Mittelpunkt. Auf diese Geschäftsmodelle wird im Folgenden nur kurz eingegangen.

Geschäftsmodell Open Source Software-Distributor
Die Komponenten einer Open Source Software werden von Distributoren auf Datenträgern zusammengefasst und können dadurch als Gesamtprodukt eingesetzt werden. Dieses Geschäftsmodell umfasst die Entwicklung, Vermarktung, Vertrieb und Support der Distributionen. OSS-Distributionen sind etwa Linux, OpenBSD oder FreeBSD. Linux-Distributionen werden z. B. von SuSE (D), Red Hat (USA), Turbolinux (USA/Japan), Caldera (USA) oder Mandrake (F) vertrieben[98].

Zielgruppen sind Unternehmer und Privatpersonen. Die Unternehmensstrategie der Open-Source-Distributoren wird als „Branding-Strategie“ bezeichnet, d. h. ihr Ziel ist der Aufbau einer hochwertigen Marke. Die Distributionen werden direkt über Internet verkauft bzw. über Presse- und Einzelhandelskanäle oder über Partnerschaften mit Computerherstellern vertrieben. Einnahmen werden ausschließlich durch den Verkauf von Distributionen auf Datenträgern über Partner an die Endkunden erzielt[99].

Geschäftsmodell Open Source Software-Applikationsanbieter
Der Markt von Open-Source-Applikationsanbieter umfasst alle Arten von Betriebssystemen, Hardwareplattformen und Anwendungszwecke. Die Vermarktung der Applikationen wird von der Community unterstützt. Die Software wird im Internet zur Verfügung gestellt und auch die Produkte werden via Internet vertrieben. Open-Source-Applikationsanbieter geben die Software entweder zur Gänze unter einer offenen Lizenz frei, oder versuchen durch das Anbieten verschiedener Lizenzmodelle die Software teilweise kommerziell zu vertreiben[100].

Geschäftsmodell Open Source Software-Appliance[101]-Hersteller
Zusätzlich zum Open-Source-Betriebssystem stellen die Anbieter spezielle Applikationen für die Schnittstellen zum Benutzer her (z. B. Administration, Updatefunktion etc.). Diese Applikationen

[98] Vgl. Leiteritz, 2004, S. 141.
[99] Vgl. ebenda, S. 142 f.
[100] Vgl. Leiteritz, 2004, S. 146 ff.
[101] Appliances sind Geräte, die aus einer Hardware-Software-Kombination bestehen.

werden häufig proprietär vertrieben. Da die Software jedoch nicht logisch bzw. technisch aufeinander aufbaut, entsteht kein Widerspruch zur GPL. Appliance-Anbieter sind beispielsweise Cobalt/Sun, IBM Whistle, Linware oder Nokia. Zielgruppe sind Unternehmen und Privatpersonen, die die Funktionen der Appliances benötigen, diese aber nicht selbst installieren können bzw. wollen. Der Vertrieb und der Kunden-Support werden dabei über Partner abgewickelt. Die Vermarktung der Produkte und nicht die des Herstellers steht im Vordergrund. Einnahmen werden durch den Verkauf der Produkte über Partner an die Endkunden erzielt, weitere Erlöse ergeben sich aus Wartungsverträgen oder Support-Anfragen der Partner[102].

1.4.2 Dienstleistungs-Geschäftsmodell

Anders als bei den Produkt-Geschäftsmodellen wird hier kein eigenes Produkt entwickelt, sondern Dienstleistungen für bereits bestehende Produkte werden angeboten. Open-Source-Dienstleister übernehmen Qualitätssicherungsfunktionen für die Komponenten, d. h. sie müssen diese überprüfen, korrigieren und mit den Entwicklern in Kontakt stehen.

Grundsätzlich können vier Gruppen von Open-Source-Dienstleistern unterschieden werden: Distributoren, große Hardware-Hersteller, globale Systemintegratoren und spezialisierte Open-Source-Dienstleister[103].

Die Zielgruppe für Dienstleister sind IT-Entscheider in Unternehmen, keine Privatpersonen. Der Markt für Support-Produkte umfasst den Installations-Support, Support-Pakete, Wartungsverträge und Integrationsleistungen. Der Vertrieb der Dienstleistungen ist ein Direktvertrieb mit direkter Kundenansprache durch die entsprechenden Mitarbeiter. Einnahmen werden durch den Verkauf von Personensätzen erzielt[104].

1.4.3 Mediator-Geschäftsmodell

Mediatoren versammeln unterschiedliche Interessengruppen im Bereich von Open Source Software auf einem so genannten Marktplatz. Der bekannteste Mediator ist SourceForge[105]. Zielgruppe von Mediatoren sind Entwickler, Nutzer, Dienstleister und Werbetreibende. Das Angebot für Entwickler umfasst Verwaltung, Hosting und Vermarktung von freien Projekten. Für Nutzer bietet der Marktplatz eine zentrale Anlaufstelle für Recherchen sowie eine Auswahl verschiedener Komponenten. Sponsoren und Werbetreibende nutzen den virtuellen Marktplatz für eine zielgruppenspezifische Werbung und für Open-Source-Dienstleister wird die Vermarktung und der Vertrieb der Dienstleistungen angeboten. Nutzer und Entwickler können den Mediator kostenlos nutzen, im Gegensatz dazu müssen Werbetreibende und Dienstleister teilweise einen Beitrag für die Teilnahme zahlen[106].

[102] Vgl. Leiteritz, 2004, S. 150-152.
[103] Mehr dazu: Leiteritz, 2004, S. 153.
[104] Vgl. Leiteritz, 2004, S. 152-155.
[105] Siehe Kapitel 1.2.4.
[106] Vgl. Leiteritz, 2004, S. 156.

Literaturverzeichnis

Grassmuck, Volker: Freie Software/Geschichte, Dynamiken und gesellschaftliche Bezüge, Version 1.0, Berlin, September 2000.

Leiteritz, Raphael: Open-Source-Geschäftsmodelle, in: Gehring, Robert A. / Lutterbeck, Bernd (Hrsg.): Open-Source-Jahrbuch 2004/Zwischen Softwareentwicklung und Gesellschaftsmodell, Berlin, 2004, S. 139-170.

O´Sullivan, Maureen u. a.: Open Source in Public Administration, 2004.

O´Sullivan, Maureen: Three Roses Report on FLOSS Law: Licensing and Legislation with the GNU GPL, 2003.

Wong, Kenneth / Sayo, Peth: Free/Open Source Software/A General Introduction.

Internetquellen

Jaeger, Till: Einmal GPL, immer GPL?/Nebeneinander von Lizenztypen, erschienen im Linux-Magazin 01/2001, Online im WWW unter URL: http://www.linux-magazin.de/Artikel/ausgabe/2001/01/recht/recht.html [Stand: 24.09.2004]

Linauer, Udo u. a.: Open Source Software als vielversprechende Alternative/Impulsreferat, Mai 2002, Online im WWW unter URL: http://www.cio.gv.at/alternativen/ImpulsReferat.pdf

http://www.gnu.org/
http://www.fsf.org
http://www.fsc.cc/
http://www.opensource.org
http://www.sourceforge.net
http://www.fact-index.com
http://www.nimm-linux.tk

Abbildungsverzeichnis

Kapitel 2

SWOT – BETRACHTUNG EINES OPEN SOURCE SOFTWARE-EINSATZES IN DER ÖFFENTLICHEN VERWALTUNG

Die – teilweise auch öffentlich geführte – Diskussion über den Einsatz von Open Source Software ist stark geprägt von weltanschaulichen Einstellungen der Befürworter auf der einen Seite und der Gegner auf der anderen Seite. Die ausgeprägte Polarisierung führt häufig zu einer emotional geleiteten Argumentation der Vor- und Nachteile von Open Source Software und fokussiert darüber hinaus häufig auf den Marktführer im Softwarebereich, nämlich Microsoft.

Um sich ein realitätsnahes Bild von den tatsächlichen Stärken und Schwächen von Open Source Software machen zu können, bedarf es daher einer fundierten Kenntnis der Thematik, die allerdings sehr stark technologielastig ist und daher Nicht-Technikern den Zugang zusätzlich erschwert. Gerade eine Argumentation anhand objektiv nachvollziehbarer Kriterien ist aber die Entscheidungsgrundlage für Führungskräfte – unabhängig, ob in der Wirtschaft oder der Verwaltung – ob ein Engagement im Bereich Open Source Software für die eigene Organisation in Frage kommt.

Mit nachfolgendem Kapitel soll genau dieses Defizit behoben werden, indem auf Grundlage einer SWOT-Analyse[1] die Stärken und Schwächen eines Open Source Software-Einsatzes objektiv und unter weitgehender Vermeidung technischer Fachterminologie dargestellt werden. Daraus abgeleitet resultieren auch die Chancen bzw. Risiken, welche sich aus Open Source Software-Projekten heraus ergeben können.

Als Zusatznutzen bietet diese Darstellungsform eine Gegenüberstellung der Charakteristika von Open Source Software und proprietärer Software[2], da sich die Stärken und Schwächen von Open Source Software natürlich an den Eigenschaften der proprietären Software messen.

Die Darstellung der Stärken und Schwächen erfolgt weiters unter dem Gesichtspunkt eines Einsatzes in der öffentlichen Verwaltung, sofern Unterschiede gegenüber einer Verwendung in anderen Wirtschaftszweigen bestehen.

2.1 STÄRKEN VON OPEN SOURCE SOFTWARE

Open Source Software unterliegt einer vollkommen anderen Philosophie als kommerziell erstellte Software. Aus diesem Umstand heraus ergeben sich zahlreiche Unterschiede, die teilweise als Stärken, teilweise jedoch auch als Schwächen zu werten sind.

2.1.1 Verfügbarkeit des Quellcodes

Einer der wesentlichsten Unterschiede resultiert aus der Verfügbarkeit des Quellcodes. Jede Software wird in einer bestimmten Programmiersprache erstellt und besteht im Grunde aus einigen – in Abhängigkeit von der Komplexität und dem Leistungsumfang der Software – bis zu zig-tausenden Seiten an Programmcode, der die Funktionsweise der Software bestimmt. Dieser Programmcode – auch Quellcode genannt – ist das eigentliche Kapital des bzw. der Verfasser

[1] Bei der SWOT-Analyse handelt es sich um eine Untersuchung der Stärken, Schwächen, Chancen und Risiken eines Themengebietes.

[2] Proprietäre Software wird auch häufig als „geschlossene Software“ oder „closed code“ oder einfach nur als „kommerzielle Software“ bezeichnet.

und wird daher bei kommerzieller Software gegen eine Einsichtnahme sowie gegen Veränderungen geschützt. Diesen Schutz erzielt man durch das so genannte „Kompilieren", was nichts anderes bedeutet als eine Übersetzung des geschriebenen Programmcodes in eine für den Computer direkt und daher wesentlich rascher ausführbare Form. Eine auf diese Weise fixierte Software ist grundsätzlich[3] weder veränder- noch einsehbar, sondern stellt ausschließlich die vorgesehenen Funktionen und Abläufe zur Verfügung. Jede notwendige Änderung kann nur vom Eigentümer des originalen Quellcodes vorgenommen werden und nach jeder Änderung ist auch wieder eine Kompilierung erforderlich. Der originale Quellcode selbst wird von den Herstellern unter strengem Verschluss gehalten.

Open Source Software unterscheidet sich von kommerzieller Software dadurch, dass der Quellcode frei verfügbar ist, von jedermann eingesehen und analysiert und natürlich auch verändert werden kann. Der Grad der Veränderungen bzw. der Softwarestatus nach erfolgter Veränderung (z. B. wer ist Eigentümer/Urheber der abgeänderten Software) wird durch verschiedene Open Source Software-Lizenzmodelle geregelt.

2.1.2 Analysierbarkeit des Quellcodes

Da der Quellcode frei verfügbar ist, kann er von jedem kundigen Programmierer analysiert werden, d. h. es ist – eine gute Dokumentation des Codes vorausgesetzt – möglich nachzuvollziehen, welche Routinen bei einer Software im Hintergrund ablaufen, welche „Schalter" welche Wirkungen verursachen, welche Programmteile für welche Funktionen zuständig sind usw. Sollten in einem Programm Fehler auftreten, so hat im Prinzip jeder Anwender die Möglichkeit – entsprechende Kenntnisse vorausgesetzt – , diese zu beheben.

Die Analysierbarkeit des Quellcodes zieht eine Reihe interessanter Optionen nach sich: Während man als Anwender bei proprietärer Software auf die so genannte „Release-Politik"[4] des Herstellers angewiesen ist, also auf dessen Fehlerbehebungs- und Verbesserungszyklen, lassen sich bei Open Source Software-Eingriffe individuell vornehmen und somit im Idealfall sehr kurzfristig umsetzen. Das Auffinden von Fehlfunktionen wird bei Open Source Software aufgrund des modularen Aufbaus zusätzlich erleichtert, da sich das Modul, welches fehlerbehaftet ist, relativ leicht identifizieren und entweder korrigieren oder durch ein anderes Modul mit gleichen oder ähnlichen Funktionen ersetzen lässt. Generell ist festzuhalten, dass die Modularität von Open Source Software eine stärkere Individualisierung in Bezug auf den benötigten Funktionsumfang zulässt.

---

[3] grundsätzlich deshalb, weil Computerexperten durchaus – allerdings sehr aufwendige und nicht erfolgssichere – Methoden zur Verfügung stehen, aus einer bereits kompilierten Software Teile des Programmcodes rückzurechnen. Man nennt diesen – an sich illegalen – Vorgang „reverse engineering".

[4] Unter der „Release-Politik" versteht man einerseits die Strategie eines Softwareherstellers, sein Produkt in eine bestimmte Richtung zu entwickeln und neue Versionen auf den Markt zu bringen. Die Release-Politik umfasst aber auch Anpassungen und Erweiterungen der bestehenden Software, sowohl aufgrund von Fehlfunktionen als auch aufgrund veränderter Rahmenbedingungen wie z. B. neuer Hardwareprodukte oder auch Software. Die „Release-Intervalle" liegen im Ermessen des Herstellers und sind ein Qualitätskriterium. Üblicherweise sammeln Hersteller Fehlerhinweise und Verbesserungsvorschläge – z. B. aus Benutzergruppen – und arbeiten diese blockweise ein. Inwieweit auf Fehler und Verbesserungsvorschläge reagiert wird, liegt allerdings ebenfalls im Ermessen des Herstellers und schafft damit einen gewissen Grad an Abhängigkeit seitens der Anwender.

2.1.3 Erweiterbarkeit von Open Source Software

Der eben erwähnte modulare Aufbau und das Vorliegen des Quellcodes ermöglichen es Nutzern, individuelle Erweiterungen zu programmieren. Diese Erweiterungen können entweder Funktionen des Betriebssystemkerns (Kernel) umfassen, in den meisten Fällen wird es sich aber um fachspezifische Funktionalitäten handeln, also um Veränderungen an einem Softwaremodul. Damit entsteht eine – gerade für die öffentliche Verwaltung mit ihren sehr spezifischen und von gesetzlichen Regelungen determinierten Anforderungsprofilen – interessante Option: Bereits vorhandene Open Source Software-Module, die für andere Wirtschaftsbereiche entwickelt wurden, können als Ausgangsbasis herangezogen, analysiert und bedarfsgerecht angepasst werden, wobei diese Anpassungen sowohl Veränderungen als auch Erweiterungen sein können. Des weiteren ist es aufgrund des modularen Aufbaus natürlich grundsätzlich möglich, komplett neue, zusätzliche Module zu erstellen und in ein bestehendes Open Source Software-Paket zu integrieren.

2.1.4 Bedarfsorientierte Funktionalität

Proprietäre Software ist einfach gesagt „Software von der Stange“ – mit allen Vorteilen und Nachteilen. Zu den Vorteilen zählt sicherlich die vergleichsweise einfache Inbetriebnahme und Anwendung – also Nutzung – der Software. Dafür bietet sie einen bestimmten, vom Hersteller definierten Rahmen an Funktionen. Wird eine Funktionserweiterung benötigt, was gerade im Bereich der öffentlichen Verwaltung relativ häufig vorkommt, so können diese – wenn überhaupt – häufig nur vom Hersteller der Basissoftware selbst vorgenommen werden. Hinzu kommt, dass individuelle Anpassungen und Erweiterungen mit relativ hohen Kosten verbunden sind und für den Bedarfsträger – also den Auftraggeber – eigentlich keine Möglichkeit einer Ausschreibung der Leistungen besteht, er somit an den Softwarehersteller gebunden ist.

Diese Umstände haben in der Vergangenheit immer wieder zur unbefriedigenden Situation geführt, dass sich die Verwaltung an die Software anzupassen hatte und nicht umgekehrt. Hinzu kommt auch, dass sich die Bereitschaft zu individuellen Veränderungen und Anpassungen bei proprietärer Software verkehrt proportional zur Position des Herstellers am Markt verhält, was jedoch aus ökonomischer Sicht auch verständlich ist: Je mehr Kunden ein Softwareprodukt einsetzen, desto weniger kann der Hersteller auf die Wünsche jedes einzelnen Kunden eingehen und muss bestrebt sein, bei seiner Releasepolitik einen Mittelweg zwischen den verschiedenen Anforderungen zu gehen.

Bei Open Source Software hat im Prinzip jeder Anwender die Chance, die von ihm benötigten Funktionalitäten bedarfsgerecht erstellen zu lassen. Dem gegenüber stehen aber der Aufwand und die Kosten für eine Realisierung. Wirklich attraktiv ist vielmehr das grundsätzliche Vorhandensein der Möglichkeit, kleinere benötigte Modifikationen schnell realisieren zu können. Bei all diesen Überlegungen darf jedoch nicht außer Acht gelassen werden, dass jede Individualisierung auch bei Open Source Software einerseits mit Kosten verbunden ist, andererseits aber auch Folgewirkungen auf den Support – also spätere Unterstützungsleistungen – hat.

Es ist eine Frage des tatsächlichen Bedarfs, der notwendigen Anforderungen, aber auch der Rahmenbedingungen und Ziele, welche Entscheidung getroffen wird: Die in der Fachszene kursierende Floskel „buy versus build“ ist das Synonym für diese grundlegend unterschiedliche Philosophie. Ganz anders stellt sich die Lage dar, wenn sich mehrere Gebietskörperschaften zusammenfinden würden, um gemeinsam und bedarfsgesteuert Software entwickeln (lassen würden) – ganz im Sinne einer übergreifenden Verwaltungskooperation[5].

2.1.5 Beständigkeit und Investitionssicherheit

Kommerzielle Software unterliegt ausschließlich der Entwicklungsstrategie des Herstellers. Käufer kommerzieller Software erwerben ein Produkt „out of the box“ und – je nach Supportbedingungen – ein bestimmtes Maß an Produktunterstützung. Diese umfasst eine Hilfestellung bei der Anwendung, Problemlösung bei technischen Schwierigkeiten und eine Anpassung an neue Technologien. Die Fehlerbereinigung (Bereitstellen von Patches oder Updates) und die Anpassungsintervalle (Ausgabe neuer Releases) liegt im Ermessen des Herstellers. Darüber hinaus gewähren Hersteller häufig Preisnachlässe bei Upgrades, also dem Umstieg auf eine aktuellere Version des gleichen Produkts oder auch bei Zukauf eines anderen Produkts aus gleichem Hause. Eine andere Supportstrategie, die beispielsweise von Microsoft verstärkt verfolgt wird, ist der Abschluss von so genannten „Enterprise Agreement“-Verträgen, bei denen laufend (z. B. jährlich) Kosten anfallen und der Vertragspartner im Gegenzug einen bestimmten Supportlevel zugesichert bekommt und vor allem immer mit der neuesten Produktgeneration versorgt wird.

Die Beständigkeit und auch die Investitionssicherheit kommerzieller Software können durchaus als Risikofaktoren gesehen werden. Im Prinzip ist ein Softwarehersteller nicht verpflichtet, ein verkauftes Produkt nach Ablauf der vertraglich zugesagten Supportfristen weiter zu unterstützen und könnte dieses entweder über einen Zeitraum hinweg auslaufen lassen oder auch sofort einstellen. Bei „Enterprise Agreement“-Verträgen ist die Gefahr einer Supporteinstellung geringer, da während der Vertragslaufzeit dem Kunden ohnehin laufend die aktuellsten Programmversionen zur Verfügung gestellt werden. Allerdings darf bei „Enterprise Agreements“ im Gegenzug nicht der Aufwand der laufenden Softwareaktualisierung (organisatorischer und Schulungsaufwand) übersehen werden. Eine Produkteinstellung in der Form, dass der Softwarehersteller seine „Enterprise Agreement“-Verträge ohne Verlängerung auslaufen lässt, ist zwar unwahrscheinlich, wäre aber im Falle von Übernahmen und Fusionen durchaus denkbar – wenn nämlich ganze Produktlinien aufgelassen werden oder in andere Produkte einfließen.

Gerade in der schnelllebigen IT-Branche sind Unternehmensübernahmen oder -zusammenschlüsse an der Tagesordnung. Eine Investitionssicherheit dahingehend ist auch bei der Auswahl von Produkten großer, internationaler Hersteller nicht gegeben.

Open Source Software bietet in Bezug auf Beständigkeit und Investitionssicherheit den großen Vorteil der Herstellerunabhängigkeit, d. h. die Software unterliegt keiner Unternehmensstrategie und kann auch nicht einfach eingestellt werden. Bei Open Source Software ist eine ständige An-

[5] siehe auch Kapitel 2.3.5

passung an neue Entwicklungen möglich und kann – sofern dies nicht ohnehin im Rahmen einer Entwicklungs-Community[6] geschieht – individuell vorgenommen werden. Aufgrund des modularen Aufbaus ist auch der Austausch einzelner Module oder des gesamten Betriebssystemkerns gegen aktuellere Versionen möglich. Auf diese Weise kann der Lebenszyklus einer Software deutlich verlängert werden bzw. ist dieser wesentlich besser planbar. Investitionssicherheit ist dahingehend gegeben, dass einmal getätigte Investitionen in die Entwicklung Open Source basierter Software durch die laufende Anpassung und Weiterentwicklung (und damit fortlaufende Nutzbarkeit) nicht verloren gehen.

Allerdings ergeben sich bei Open Source Software gerade in Bezug auf Support und Weiterentwicklung anderweitig Probleme, die ebenfalls Unsicherheitsfaktoren darstellen (siehe Schwächen/ Risiken von Open Source Software).

2.1.6 Unabhängigkeit

Da Open Source Software frei ist, besteht keine Abhängigkeit von einem Hersteller, von dessen Produkt(entwicklungs)politik und von dessen Support. Die gewonnene Unabhängigkeit ist allerdings trügerisch und sollte nicht überbewertet werden. Sie trifft vielmehr dann zu, wenn Open Source Software selbst – also organisationsintern oder im Rahmen einer unter 2.1.5 beschriebenen Verwaltungskooperation – erstellt wird. Erwirbt man eine so genannte Distribution – also eine Zusammenstellung ausgewählter Open Source Software-Module inklusive distributorenspezifischer Erweiterungen – , so begibt man sich bis zu einem bestimmten Grad in ein Abhängigkeitsverhältnis zum Distributor. Nur zu einem bestimmten Grad deshalb, da bei Open Source Software im Gegensatz zu proprietärer Software einzelne Module ersetzt oder Programmteile durch herstellerfremde Programmierer verändert werden können. Dennoch muss man sich bewusst sein, dass ein Umstieg von einer Distribution auf eine andere oder ein Eingriff in die Zusammenstellung einer Distribution vor allem in der flächendeckenden organisationsinternen Integration einen beträchtlichen Aufwand verursacht. Dies gilt vor allem für spezielle Serverpakete oder für Spezialdistributionen, die auf einen gemischten Betrieb von Open Source Software und Microsoft-Produkten ausgelegt sind.

Bei Open Source Software, die organisationsintern oder im Auftrag erstellt wird, besteht ebenfalls die Gefahr einer Abhängigkeit, nämlich von den hauseigenen Programmierern oder vom Auftragnehmer, der die Softwareentwicklung durchgeführt hat. Auch wenn der Quellcode eines Programms noch so gut dokumentiert wurde – was in der Praxis eher die Ausnahme und nicht die Regel darstellt – , so verursacht das Einarbeiten in ein fremdes Programm – z. B. im Falle eines Programmierer- oder Vertragspartnerwechsels – einen ziemlich hohen (und damit kostenintensiven) Aufwand.

Trotz dieser Gefahren besteht bei Open Source Software durch die freie Verfügbarkeit des Quellcodes faktisch Unabhängigkeit von Herstellern und Supportgebern.

[6] siehe Kapitel 1 (Grundlagen) – Open Source Software-Landschaft und -Strukturen

2.1.7 Sicherheit, Qualität und Stabilität der Software

Auf den ersten Blick mag es widersprüchlich wirken – dass nämlich Open Source Software, bei der der Quellcode frei verfügbar ist und damit auch manipular wäre, sicherer sein soll als proprietäre Software, auf deren Quellcode niemand bis auf den Hersteller Zugriff hat und daher auch durch herstellerfremde Personen keine Veränderungen möglich sind.

In der Praxis hat sich aber genau das Gegenteil erwiesen: Die größten Sicherheitslücken liegen bei proprietärer Computersoftware, wohingegen Open Source Software relativ hohen Sicherheitsanforderungen gerecht wird und in der Vergangenheit auch kaum Angriffsziel von Computerviren und unerwünschten Systemzugriffen oder -veränderungen wurde. Für diesen – oberflächlich betrachtet – widersprüchlichen Zustand gibt es mehrere Gründe:

Kommerzielle Software unterliegt den Bedingungen des Marktes und es liegt daher im Interesse der Hersteller, die Erstellungskosten möglichst gering zu halten. Auch der Zeitfaktor spielt eine wesentliche Rolle, sodass die Qualitätssicherung zu kurz kommt. Außerdem hat es sich im Softwaremarkt durchwegs eingebürgert, die Fehlersuche auf den Kunden (zumindest auf die Gruppe der „early adopters“) zu überwälzen. Ein weiterer Faktor ist die Genesis kommerzieller Software, die sich häufig über Jahre hinweg entwickelt hat, an den Bedarf des Marktes angepasst und erweitert wurde. Dementsprechend unübersichtlich, heterogen und häufig im Kern auf überkommene Techniken aufbauend präsentiert sich der Quellcode. Oftmals wäre eine komplette Neuentwicklung am sinnvollsten, aus Kostengründen muss der alte Quellcode aber weiter verwendet und angepasst werden. Das Ergebnis sind Sicherheitslücken, Qualitätsmängel und Kompatibilitätsprobleme.

Der Quellcode von Open Source Software ist zwar für jedermann einsehbar, die Entwicklung – insbesondere größerer Softwareprojekte – unterliegt aber bestimmten Regeln, deren Einhaltung von einer Community (Open Source Software-Entwicklungsgruppe) beachtet wird. Auch bei Open Source Software kann nicht einfach ein korrupter Programmierer Hintertüren oder Sicherheitslücken in den Quellcode des Softwareprojekts einbauen, sondern die Mitglieder der Community liefern ihre ausprogrammierten Beiträge (z. B. Codeteile für in sich geschlossene Funktionalitäten) ab und diese werden von einem Supervisor (= Maintainer) oder einer Gruppe begutachtet und erst danach freigegeben. Da bei größeren Projekten eine Vielzahl von Programmierern freiwillig mitarbeiten, werden Veränderungen am Quellcode laufend von einer großen Gruppe von Experten validiert und würden Fehler oder absichtlich eingefügte, schädliche Codes sehr rasch entdeckt und beseitigt werden.

Die Validierung von Open Source Software erfolgt permanent und von einer sehr großen Zahl an Experten, sodass die Wahrscheinlichkeit, dass Fehler entdeckt und rasch behoben werden, sehr hoch ist. In der Praxis sind tatsächlich die Zeitintervalle vom Aufspüren eines Fehlers bis zur Lieferung eines geeigneten Patches – das sind kleine Programmroutinen zur Behebung spezieller Fehler – bei Open Source Software wesentlich kürzer als bei proprietärer Software. Bei kommerzieller Software wäre eine derart umfangreiche Qualitätssicherung aus Kostengründen einfach nicht machbar.

Ein weiterer Faktor, der eine höhere Sicherheit von Open Source Software erlaubt, ist die bereits bekannte Modularität derselben. Durch so genanntes „Hardening“ – darunter versteht man das „Abspecken“ einer Software auf den gerade noch benötigten Funktionsumfang – mindert man auch das Risiko von Schwachstellen und damit einer Anfälligkeit auf Angriffe. Benötigt man beispielsweise auf einem Computer ausschließlich eine Firewall, so ist es nicht notwendig, mit der Software auch andere Internetfunktionalitäten wie z. B. einen Webserver zu installieren. Bei proprietärer Software ist dies nicht oder nur im Rahmen der vorgegebenen Möglichkeiten durch den Hersteller gegeben.

2.1.8 Modularität

Der modulhafte Aufbau von Open Source Software ist schon vielfach erwähnt worden. Es ist ein herausragendes Charakteristikum, dass sich ein Open-Source-Programm in Wahrheit aus mehreren Teilpaketen mit klar abgegrenztem Funktionsumfang zusammensetzt. Module können ausgetauscht und ersetzt werden, abgespeckt – also ohne Konsequenzen für die Lauffähigkeit des Grundprogramms entfernt werden. Bei Problemen in einem Modul gibt es kaum einen Totalabsturz des Systems, sondern es wird nur das betroffene Modul beendet. Dadurch lassen sich Fehler besser eingrenzen und lösen. Aufgrund der Modularität sind auch viele verschiedene Zusammenstellungen möglich, wie sich in den verschiedensten Distributionen zeigt.

2.1.9 Offene Standards – Kompatibilität

In der Open-Source-Welt gibt es wenig Anlass, Funktionsweisen und Schnittstellen von Software zu verheimlichen. Vielmehr ist es den beteiligten Open-Source-Programmierern häufig ein Anliegen, dass „ihre“ Software kompatibel zu möglichst vielen anderen Produkten ist. Gewährleistet wird dies durch die Einhaltung von Standards – so genannten offenen Standards – , die sich in der Open-Source-Szene etabliert haben und deren Definition vollkommen frei zugänglich ist.

Aufgrund des Drucks seitens der Anwender gehen auch vermehrt Hersteller kommerzieller Software dazu über, offene Standards in ihre Produkte zu implementieren, um ein Zusammenspiel von Open Source Software und kommerzieller Software zu verbessern und damit einen Marktvorteil zu erlangen.

2.1.10 Einsatz in heterogenen Systemlandschaften

Die Entwicklung von Software, die heute gemeinhin als Open Source verstanden wird, also auf Linux als Betriebssystem basierend, hat relativ spät, nämlich erst in den 90er Jahren des vergangenen Jahrhunderts, eingesetzt. Zu diesem Zeitpunkt waren bereits Systeme wie Unix, Windows, Solaris u.Ä. am Markt und die Entwickler von Linux bzw. anderer Open Source Software auf Linux- Basis konnten sich daran orientieren. Da die Offenheit gegenüber und Kompatibilität zu anderen Produkten eine der Grundphilosophien des Open-Source-Gedanken darstellt, wurde und wird bei den Programmierungen einerseits auf Kompatibilität, andererseits auch auf Multiplattformfähigkeit gesetzt. Viele Open-Source-Programme sind für mehrere Betriebssysteme verfügbar (z. B. OpenOffice), laufen überhaupt plattformunabhängig, wie dies im Falle von browserbasierten Anwendungen der Fall ist. Darüber hinaus gibt es bereits mehrere Open-Source-

Anwendungen, die einen Doppelbetrieb – nämlich z. B. Linux- und Windows-Programme – auf einem Computer erlauben, indem eine andere Plattform – eben beispielsweise Windows für Windows-Programme – simuliert wird. Es handelt sich dabei um so genannte Emulationen[7].

Die Offenheit und variable Einsatzmöglichkeit von Open Source Software in einer heterogenen Systemlandschaft begünstigt einen Umstieg, da dieser nicht sofort und vollständig erfolgen muss, sondern in Form verschiedener Umstiegsszenarien Schritt für Schritt vorgenommen werden kann und auch eine Koexistenz mit den bestehenden Systemen möglich ist. So bestünde beispielsweise die Möglichkeit, eine Office-Suite (Textverarbeitung, Tabellenkalkulation, Präsentationserstellung), die unter einer Open-Source-Lizenz läuft (wie z. B. OpenOffice), auf einem Windows-Client (also einem herkömmlichen Windows-PC) zu installieren und zu betreiben.

Obwohl die Entwickler von Open Source Software grundsätzlich nach Kompatibilität trachten, sind Probleme in heterogenen Systemlandschaften – wie z. B. bei den Dateiformaten – aber dennoch nicht auszuschließen. Dieser Problematik muss man sich sowohl bei einem teilweisen, als auch bei einem Totalumstieg bewusst sein. Auch wenn verwaltungsintern im Falle einer kompletten Umstellung keine Kompatibilitätsprobleme bestehen, so können diese bei verwaltungsfremden Dateien nicht vermieden werden.

Die Ursache liegt vor allem darin begründet, dass die Programmierer von Open Source Software ständig den Neuerungen anderer Software- und Hardwarehersteller – allen voran Microsoft – nachzuziehen haben, um wieder Kompatibilität zu erreichen. Eine weitere Ursache ist teilweise auch mangelndes Interesse der Hersteller kommerzieller Software, mit ihren Produkten kompatibel zu Open Source Software zu sein.

2.1.11 Rechte und Lizenzen

Die offene Lizenzpolitik ist wohl eine der größten Stärken von Open Source Software und gleichzeitig insbesondere für die öffentliche Verwaltung auch mit gewissen Risiken behaftet. Auf die unterschiedlichen Lizenzierungsvarianten wurde bereits im Grundlagenkapitel ausführlich eingegangen. Die Rechtslage beim Einsatz von Open Source Software wird darüber hinaus aufgrund der großen Bedeutung für die öffentliche Verwaltung in einem eigenen Autorenbeitrag abgehandelt. Ein Grundprinzip von Open Source Software ist die Veränderbarkeit und unbeschränkte Weitergabe des Quellcodes, was an sich im Widerspruch zu urheberrechtlichen Regelungen steht. Open-Source-Programmierer treten – indem sie ihren Programmcode freiwillig unter eine bestimmte Lizenz stellen – ihre Rechte ab. Das Problem liegt vielmehr in der rechtlichen Haltbarkeit der diversen Open-Source-Lizenzen vor Gericht, da die Lizenzierungsautoritäten nicht staatlich sind, sondern aus freiwilligen Vereinigungen und ähnlichen, eher „weichen“ Organisationsstrukturen hervorgegangen sind. Ein Paradefall ist beispielsweise der seit Jahren laufende Rechtsstreit eines US-Unternehmens (SCO), das behauptet, Teile des Linux-Kernels würden urheberrechtswidrig aus lizenziertem Unix-Quellcode dieses Unternehmens bestehen. Allerdings gibt es – auch im deutschsprachigen Raum – bereits gerichtliche Musterentscheidungen, bei denen die Gültigkeit von Open-Source-Lizenzen bestätigt wurde.

[7] Bekannte Windows-Emulationen, die Windows-Programmen eine entsprechende Windows-Umgebung vorgaukeln, sind beispielsweise die Programme „wine“ oder „vmware“.

2.1.12 Kostenstruktur, Marketing- und Overheadkosten

Ein entscheidender Vorteil von Open Source Software sind sicherlich die niedrigeren Anschaffungskosten. Es ist eine Illusion zu glauben, Open-Source-Programme wären kostenlos. Der Aufwand, selbst die notwendigen Module zusammenzustellen, würde rasch die Kosten der Anschaffung einer Distribution, also einer fertigen Zusammenstellung, übersteigen. Beim Einsatz von Open Source Software kommt es vielmehr zu einer Kostenverlagerung von Sachaufwand in Form von Lizenzkosten hin zu Personalkosten.

Bestehende Open-Source-Produkte – also bereits entwickelte und weitgehend ausgereifte Produkte wie z. B. OpenOffice – werden von einer großen Programmierer-Community gepflegt, hier entstehen keine Kosten. Die Programmbasis wird von Unternehmen herangezogen und benutzerfreundlicher (z. B. bei der Installation) gestaltet, mit umfangreicheren Hilfefunktionen oder erweiterten Funktionalitäten ausgestattet. Diese Verbesserungen verursachen Kosten, die über den Verkauf solcher „kundenfreundlicherer" Zusammenstellungen hereingespielt werden müssen. Da nur Anpassungen notwendig sind – die Kernprogrammierleistung wird ja von der Open-Source-Community erbracht – , halten sich auch die Personal- und Overheadkosten in Grenzen. Darüber hinaus wird – bzw. wurde in der Vergangenheit – von Open-Source-Lösungsanbietern weniger in Marketingaktivitäten investiert.

Ein Beispiel für diese Anpassung von Open Source Software ist das bereits mehrmals erwähnte Office-Paket „OpenOffice". OpenOffice wird von einer Open-Source-Community entwickelt und kann kostenlos über das Internet heruntergeladen werden. Die Firma SUN hingegen verkauft StarOffice, das zwar auf OpenOffice basiert, allerdings mit zahlreichen Erweiterungen und Verbesserungen versehen wurde. Da der Aufwand dieser Adaptierungen um ein Vielfaches geringer ist als eine Neuprogrammierung, sind auch die Anschaffungskosten von StarOffice niedriger als bei einem proprietären Office-Produkt.

2.2 SCHWÄCHEN VON OPEN SOURCE SOFTWARE

Keine Vorteile ohne Nachteile, natürlich weist Open Source Software gegenüber proprietärer Software einige Schwächen auf, derer man sich bewusst sein sollte, wenn man eine Umstellung plant. Manche dieser Schwächen betreffen mögliche Defizite, die man durch entsprechende Vertragsgestaltung oder Absicherung vor einem Umstellungsprojekt minimieren kann, andere wiederum stellen ein Charakteristikum von Open Source Software dar und können nur akzeptiert werden. Im nachfolgenden Kapitel werden die wesentlichen Schwächen kurz dargestellt und – sofern zutreffend – auf spezielle Ausprägungen im Bereich der öffentlichen Verwaltung näher eingegangen.

2.2.1 Mangel an geeigneten/ausgereiften Produkten

Die Entstehung von vielen Open-Source-Produkten ist mehr oder weniger vom Zufall bestimmt, es gibt jedenfalls keine akkordierte Entwicklungsstrategie. Im Regelfall beginnt ein Open Source Software-Projekt mit dem Zusammenschluss einer kleinen Gruppe ambitionierter Personen mit

Programmierkenntnissen, angetrieben vom gleichen Bedarf nach einer speziellen Softwarelösung, die entweder in der gewünschten Form noch nicht am Markt verfügbar ist oder nur zu sehr hohen Preisen. Vermehrt werden Open-Source-Projekte auch von Unternehmen bis hin zu Konzernen initiiert und auch mit Personal unterstützt. Der Nutzen aus einer derartigen Vorgangsweise ist trotz der freien Verfügbarkeit des Quellcodes vielfältig: Unternehmen profitieren von der raschen Entwicklung von Lösungen durch externe Beteiligung, d. h. es ist weniger firmeneigener Ressourceneinsatz erforderlich, um dennoch zu guten Ergebnissen zu kommen. Der Umsatz wird nicht überwiegend über den Produktvertrieb, sondern stärker über Dienstleistungen – z. B. im Zuge der Softwareimplementierung – gemacht. Programme unter freier Lizenz profitieren von der Kompatibilität mit offenen Standards usw.

Der Nachteil ist, dass es sich bei den entwickelten Softwareprodukten – sei es nun auf Initiative von privaten Entwicklern oder von Unternehmen – meist um echte Nischenanwendungen – wie z. B. ein extrem abgespecktes Betriebssystem für Mobiltelefone auf Basis eines Linux-Kernels – oder um populäre Softwareprodukte wie z. B. ein Content Management System handelt, von denen es bereits unzählige am Markt gibt.

Gerade die öffentliche Verwaltung, deren Bedarf auf fachspezifischen Anwendungen wie z. B. den Baubereich, das Personenstandswesen oder das Rechnungswesen liegt, ist von fertigen oder bereits ausgereiften Open-Source-Produkten total unterversorgt. Aber auch über die verwaltungsspezifischen Anforderungen hinaus ist ein Mangel an geeigneter oder ausgereifter Software festzustellen. Für Anwender, die nicht in einer Open-Source-Community an einem Projekt mitarbeiten, ist es darüber hinaus relativ schwierig festzustellen, welchen Reifegrad ein Softwareprojekt hat und wie intensiv die Entwicklung tatsächlich voranschreitet. Sourceforge, die größte Entwicklungsplattform, die tausende Communities beherbergt, hat zwar ein Bewertungssystem eingeführt, aus dem Parameter wie Entwicklungsstand, Reifegrad, beteiligte Personen im Kernteam, Intervall der Releasesprünge usw. hervorgehen, zuverlässige Indikatoren für die zukünftige Entwicklung sind diese jedoch keine.

Generell kann festgehalten werden, dass mehr Anwendungen mit hohem Reifegrad und hoher Stabilität auf Open-Source-Basis verfügbar sind, je unspezifischer das Einsatzgebiet ist. Produkte wie Webserver, Datenbanken, Kommunikationssoftware für Arbeitsgruppen, Content Management Systeme für den Aufbau von Internetplattformen und jegliche Art von Internetsoftware wie z. B. Webshops, Gästebucher, Foren usw. – also allesamt Produkte, die keiner bestimmten Branche zuordenbar sind und daher universal einsetzbar sind, sind häufig auch als ausgereifte Open Source Software verfügbar; oftmals sogar qualitativ besser (in Bezug auf Fehlerhäufigkeit und Stabilität) als vergleichbare proprietäre Software.

2.2.2 Mangelnde Hardwareunterstützung bei neuer Hardware

In die gleiche Kerbe schlägt das Problem mangelnder Hardwareunterstützung. Da es bei Open Source Software keine festgeschriebene Entwicklungsstrategie und Produktlinie gibt, ist eine Unterstützung neuer technischer Komponenten oder Geräte abhängig von den Herstellern oder von ambitionierten Open-Source-Entwicklern. Seitens der Hersteller neuer EDV-Komponenten oder Geräte besteht zwar grundsätzlich Interesse daran, dass ihre Produkte möglichst breit von

Software unterstützt werden, allerdings gibt es einige Hürden:

- Für manche Produkte ist die Gemeinde der Open-Source-Anwender teilweise noch zu klein, sodass sich eine aufwändige Softwareunterstützung nicht rechnet;

- die Open Source Software-Landschaft ist aufgrund der Modularität fragmentiert, so kann beispielsweise ein Linux-Betriebssystemkern mit vielen verschiedenen Zusatzmodulen und Konfigurationen installiert werden. Der Aufwand, diese vielfältigen Variationen auch herstellerseitig so abzudecken, dass die Hardware unter den unterschiedlichsten Konfigurationen zuverlässig funktioniert, ist relativ hoch;

- manche Hardwarehersteller stehen in Kooperation mit kommerziellen Softwarehäusern wie z. B. Microsoft und haben möglicherweise auch aus unternehmensstrategischer Sicht kein Interesse, Open Source Software technisch zu unterstützen.

Auch private Entwickler haben grundsätzlich die Möglichkeit, an einer Open Source Software Unterstützung neuer Hardware mitzuwirken. Man kann davon ausgehen, dass Programmierer und private Entwickler die Tendenz haben, technisch auf dem letzten Stand zu sein und schon aus diesem Grund oftmals gezwungen sind, sich bei mangelnder Hardwareunterstützung selbst zu helfen. Die größte Hürde dabei sind wahrscheinlich die technischen Spezifikationen und exakten Funktionsbeschreibungen, die von den Herstellern im Regelfall nicht offen gelegt werden.

Ein weiteres Problem in diesem Bereich ist das Auffinden passender Softwaremodule (Treiber) für die Unterstützung neuer Hardware, wenn diese nicht vom Hersteller selbst beigestellt werden. Für Privatpersonen mag es tragbar sein, Interessentenforen und Internetseiten nach der passenden Software abzusuchen, im Unternehmensalltag oder in der Verwaltung ist so eine Vorgangsweise wohl kaum möglich.

Dem Problem der mangelnden Hardwareunterstützung kann insofern begegnet werden, indem vor einem Umstieg auf Open Source Software die zu übernehmende Hardware bzw. ausgefallene Komponenten auf Open-Source-Tauglichkeit überprüft werden und im laufenden Verwaltungsbetrieb einerseits nicht die aktuellsten, gerade erst auf den Markt gekommenen Produkte anschafft und andererseits bereits im Vorfeld die Open Source Software-Kompatibilität sicherstellt.

2.2.3 Keine Herstellergarantie – keine Produkthaftung

Open Source Software hat häufig keinen Hersteller, sondern entstammt einer Entwicklergruppe oder der Hersteller ist aufgrund der Modularität der Software nicht eindeutig nachvollziehbar. Distributoren – also Unternehmen, die Open Source Software mit komfortabel gestalteten Installations- und Bedienungsroutinen sowie eigenen Weiterentwicklungen anreichern, haften nicht für Produkte Dritter, bieten aber im Regelfall Supportleistungen und auch Supportverträge an.

Problematisch wird die Situation eher bei schwerwiegenden Mängeln, aufgrund deren ein Schaden entstanden ist, z. B. durch Sicherheitslücken oder Programmierfehler.

Das Problem der mangelnden Herstellergarantien und Produkthaftung lässt sich in der Praxis teilweise durch gute Wartungs- oder Supportverträge abfangen. Wenn Open Source Software von einem Hersteller bezogen wird, der den Quellcode zwar unter eine freie Lizenz stellt, das fertige Produkt und vor allem Implementierungs-Know-how verkauft, stellt sich das skizzierte Problem ohnehin nicht.

2.2.4 Keine Entwicklungsgarantie

Bei Open Source Software besteht zwar nicht die Gefahr einer plötzlichen Einstellung oder Neuorientierung des Produkts, aber Entwicklungsgarantie ist ebenfalls keine gegeben. Große, etablierte Hersteller geben eine Entwicklungsgarantie, in jedem Fall besteht aber eine Supportgarantie über einen bestimmten Zeitraum. Bei kommerziellen Softwareproduzenten kann man außerdem erwarten, dass diese an einer Verbesserung und Weiterentwicklung ihrer Produkte interessiert sind, um am Markt weiter bestehen oder sogar expandieren zu können.

Da Open Source Software häufig von privaten Entwickler-Communities programmiert wird, die auch gruppendynamischen Prozessen unterliegen, kann den Fortschritt und die Richtung solcher Projekte niemand wirlich vorhersagen. In seltenen Fällen kommt es auch zu einer Aufspaltung der Software (source forking) oder sogar der Community.

Die nicht vorhandene Entwicklungsgarantie wird jedoch zu einem bestimmten Grad von der Verfügbarkeit und dem Offenlegen des Quellcodes kompensiert. Somit könnte im Ernstfall – wenn nämlich ein Open Source Software-Projekt eingestellt oder nur zäh fortgeführt wird – jeder Anwender die Software bedarfsorientiert weiterentwickeln und auch pflegen.

2.2.5 Problem bei Service und Support

In der Vergangenheit war für Unternehmen insbesondere ein gesicherter Support von Open Source Software-Produkten ein kritischer Faktor. Gerade bei unternehmenskritischen Anwendungen ist zuverlässiger und rascher Support gefragt. Bei kommerzieller Software ist Unterstützung der Kunden ein bedeutender Geschäftszweig und oftmals einträglicher als der Verkauf. Bei Open Source Software sind Service und Support aufgrund der nicht häufig vorhandenen Hersteller problematisch. Dazu kommt, dass es im Open-Source-Bereich wesentlich weniger Know-how-Träger gibt bzw. gab als bei anderen Technologien wie z. B. Windows-Produkten und Open Source Software von der Wart- und Bedienbarkeit auch höhere Anforderungen stellt.

Mit der Verbreitung von Open Source Software und zunehmender Professionalisierung durch Beteiligung kommerzieller Unternehmen haben sich vermehrt auch Dienstleister für Service und Support von Open-Source-Produkten etabliert. Mit diesen können Wartungsverträge abgeschlossen und damit der nicht vorhandene Herstellersupport kompensiert werden.

2.2.6 Problem bei Dokumentation und Leitfäden

Open-Source-Projekte sind Gruppenprojekte mit verteilten Aufgaben. Wie auch in Softwarehäusern gibt es bei der Entwicklung Teams für Teilaufgaben und ein verbindliches System für

das Einbringen neuer oder veränderter Programmcode-Teile. In einer Kerngruppe werden die Aktivitäten der Entwicklungsteams koordiniert. Nichtsdestotrotz handelt es sich um heterogene Gruppen mit Akteuren, die über sehr unterschiedliche Qualifikationen verfügen und aus verschiedensten Gründen und Motivationen in ihrer Freizeit an solchen Projekten mitentwickeln. Man kann also davon ausgehen, dass bei Open-Source-Programmierungen sehr unterschiedliche Programmierstile und auch -qualifikationen aufeinander treffen und bezüglich der Dokumentation des Codes keine – wie in Unternehmen üblich – institutionalisierte und klar geregelte Qualitätssicherung stattfindet. Diese erfolgt eher durch Zufall, indem Neuerungen im Quellcode von einer sehr großen Anzahl an Mitgliedern der Entwicklungscommunity begutachtet werden.

Zwar gibt es grundsätzliche Regeln, wie Programmcode auszuzeichnen ist, verbindlich vorschreiben kann dies den freiwilligen Mitarbeitern an einem Open Source Software-Projekt aber niemand. Unzureichend kommentierter Quellcode ist eines der Hauptprobleme von Software, da das Hauptaugenmerk auf die Programmierung und Problemlösung gelegt wird und unter zeitlichem Druck die Dokumentation sowie das nachfolgende Erstellen von Leitfäden zu kurz kommen. Auch die Hersteller proprietärer Software haben damit unternehmensintern zu kämpfen, insbesondere wenn führend an der Entwicklung beteiligte Programmierer das Unternehmen längst verlassen haben und Änderungen an „altem“ Quellcode notwendig werden.

2.2.7 Höhere Anforderungen an Systemadministration und Anwender

Open Source Software ist deutlich komplizierter und weniger benutzerfreundlich in der Administration wie auch in der Anwendung. Dieser Umstand ist einleuchtend, wenn man bedenkt, dass Open Source Software ausschließlich von IT-Experten entwickelt wurde und wird, ohne ökonomischen Druck, nicht unter dem Diktat der Zeit und vor allem ohne konkrete Vorgaben bezüglich Benutzerfreundlichkeit und Usability. Das Programmieren von Open Source Software wird von manchen Entwickler vielmehr als Mittel zur Profilierung in Fachkreisen gesehen und es gilt, je anspruchsvoller die technische Lösung, desto besser wirkt sie. Auf der anderen Seite stehen die Programmierfreaks, die sich selbst verwirklichen wollen.

Das Ergebnis ist oft ein komplizierter und fragmentierter Quellcode sowie eine Softwareverwaltung und -anwendung, die eher den Anforderungen eines Programmierers oder zumindest IT-Kundigen entspricht und weniger auf die breite Masse abzielt. Dem Problem begegnen allerdings vermehrt Distributoren und Unternehmen, die an der benutzerfreundlichen Zusammenstellung von Open-Source-Produkten zu einfach und komfortabel zu bedienenden, eventuell spezialisierten Paketen verdienen.

2.2.8 Hoher Integrationsaufwand

Der hohe Integrationsaufwand in bestehende Systeme liegt meist weniger an der zu integrierenden Open Source Software als vielmehr an Kompatibilitätsproblemen seitens der bestehenden Hard- und Softwareumgebung. Aufgrund der Modularität von Open Source Software können einzelne Programmteile ausgetauscht und durch passendere Programme ersetzt werden. Auch diese – an sich sehr sinnvolle Variabilität – verursacht höheren Integrationsaufwand, da vor einer Umstellung sämtliche Open-Source-Module vorhanden und auch kompatibel (z. B. über offene Standards) sein müssen.

Für erhöhten Aufwand sorgt auch der meist notwendige Mischbetrieb oder Übergangsbetrieb, wenn die alte Systemlandschaft aufgrund von Software, für die es keine Open-Source-Alternative gibt, weiterhin – zumindest in Teilbereichen – aufrecht erhalten bleiben muss. Je nach Umstiegsszenario kann es auch notwendig sein, beide Softwarelandschaften – also Open Source Software und kommerzielle Software – aufeinander abzustimmen oder auf einer gemeinsamen Plattform zu betreiben. Wenn auch technisch lösbar, sollte der Aufwand eines Mischbetriebs bzw. eines Übergangs auf Open-Source-Vollbetrieb nicht unterschätzt werden.

2.2.9 Umschulungsaufwand – soziale Komponente

Der mit einer Softwareumstellung verbundene Aufwand im personellen Bereich stellt sicherlich einen kritischen Faktor dar und darf ebenfalls nicht unterschätzt werden. Wie bei allen verwaltungsinternen Großprojekten, welche die Bediensteten unmittelbar betreffen, sind auch bei einer Open Source Software-Umstellung die Mitarbeiter von Anfang an in das Projekt mit einzubeziehen und über den Projektverlauf zu informieren. Vor einer Umstellung sollten folgende Probleme bedacht und Lösungsansätze erarbeitet werden:

- Die meisten Mitarbeiter sind ausschließlich mit Windows-Programmen vertraut. Auch im Privatbereich läuft auf den meisten PCs eine Windows-Installation. Eine Umstellung auf Open Source Software in der Verwaltung – insbesondere im Client-Bereich, also auf den Arbeitsplatzgeräten und bei Programmen, die häufig benützt werden wie z. B. Textverarbeitung – bedeutet für Mitarbeiter nicht nur ein Brechen mit langjährigen Gewohnheiten und einen Lernaufwand, sondern unter Umständen auch Inkompatibilität mit der privaten Softwareausstattung. Dies kann insbesondere bei Mitarbeitern in höheren Positionen, die Arbeit mit nach Hause nehmen, zu Problemen führen.

- Ältere Mitarbeiter mit geringen Computerkenntnissen bedürfen einer intensiveren Betreuung beim Umstieg, um Frustration am Arbeitsplatz zu vermeiden.

- Der Aufwand für laufenden Support wird zwangsläufig stark ansteigen, da in der ersten Phase viele Bedienungsfehler auftreten werden und „Selbsthilfe“ durch erfahrenere Kollegen oder Geschäftsbereichsadministratoren noch nicht in vollen Umfang greift. Einer der Hauptunterschiede zwischen Open-Source- und Windows-Betriebssystemen ist die Konfiguration in tieferen Systemebenen. Während bei Windows alle Einstellungen auf einer grafischen Oberfläche und menügesteuert vorgenommen werden können, stehen bei Open-Source-Betriebssystemen nur die wichtigeren Funktionen benutzerfreundlich in Form von Fenstern und Formularen zur Verfügung, tiefergehende Systemeinstellungen sind kommandozeilenbasiert vorzunehmen.

Aktuelle Open Source Software für Arbeitsplatzgeräte läuft so wie Windows auf einer grafischen Benutzeroberfläche und ist von der Bedienung vergleichbaren Windows-Programmen sehr ähnlich. Der tatsächliche Umlernaufwand ist eigentlich nicht allzu hoch, das Hauptproblem ist viel mehr die mentale Barriere und die wahrscheinlich skeptische Erwartungshaltung. Dieses Problem sollte bereits bei den Wurzeln erfasst werden, noch bevor Gerüchte auftauchen oder falsche Erwartungshaltungen aufgebaut werden, indem beispielsweise PCs mit Demoinstallationen in den Geschäftsbereichen zum freien Ausprobieren aufgestellt werden oder im Vorfeld der Um-

stellung vergleichende Vorführungen organisiert werden, indem auf zwei Präsentationswänden jeweils ein Windows-Programm und das entsprechende Open Source Software-Pendant projeziert werden und nacheinander in beiden Programmen die häufigsten Anwendungsfälle durchexerziert werden.

2.3 CHANCEN EINES OSS-EINSATZES

Der Einsatz von Open Source Software eröffnet der Verwaltung eine Reihe von Chancen, die bei einer Entscheidung für oder gegen Open Source abzuwägen sind. Die Bandbreite reicht dabei von politischen Strategien wie z. B. einer Förderung der lokalen IT-Wirtschaft über eine Flexibilisierung der Softwareanbieterstruktur zur Verbesserung der Auftraggeberposition oder dem Wunsch nach größerer Anbieterunabhängigkeit bis hin zu budgetären Überlegungen, um langfristig Kosteneinsparungen erzielen zu können.

2.3.1 Anbieterunabhängigkeit

Aufgrund der Verfügbarkeit des Quellcodes von Open Source Software kann – zumindest theoretisch – der Anbieter gewechselt werden, wobei hier eigentlich zwischen den Begriffen „Anbieter“ und „technischer Betreuer“ differenziert werden muss. Es gibt am Markt zahlreiche Anbieter von Produkten, die auf Open-Source-Basis entwickelt und vertrieben werden. Diese Produkte werden – entweder mit oder ohne Unterstützungsleistungen – erworben, der Unterschied liegt in der Verfügbarkeit des Quellcodes und der freien Verwendung. Die Produkte können daher von einem „technischen Betreuer“, der natürlich auch der Anbieter sein KANN, aber – im Gegensatz zu proprietärer Software nicht sein MUSS, auf die spezifischen Anforderungen der Verwaltung adaptiert werden. Daraus ergeben sich zwei wesentliche Vorteile:

- Es gibt im Open-Source-Bereich mehrere Anbieter und keinen marktbeherrschenden Monopolisten, was sich positiv auf den Wettbewerb und die Preisstruktur auswirkt.

- Bei einer Anpassung der Software ist die Verwaltung nicht an den Hersteller gebunden, sondern kann diese von jedem geeigneten Unternehmen durchführen lassen.

Eine Anbieterunabhängigkeit bietet der Verwaltung somit größere Flexibilität bei der Auswahl der Produkte und Anbieter, die Chance, den Wettbewerb und die freie Marktwirtschaft zu fördern und auf diesem Wege auch Einfluss auf die Preispolitik von Open Source Software-Anbietern zu nehmen.

2.3.2 Förderung der lokalen Wirtschaft

Ein ganz wesentlicher Aspekt mit volkswirtschaftlichem Hintergrund ist die Möglichkeit, durch den Einsatz von Open Source Software die lokale Wirtschaft zu unterstützen. Die meisten proprietären Produkte großer Softwarekonzerne wie z. B. Microsoft, Oracle, Apple usw. kommen aus den USA, mit einem Ankauf dieser Produkte findet – zumindest anteilig – ein Wertschöpfungsabfluss statt. Gleiches gilt für Unterstützungsleistungen, die von internationalen Konzernen in Anspruch

genommen werden. Teile der Wertschöpfung fließen in die lokalen Niederlassungen, ein durchaus beträchtlicher Anteil geht an die Stammhäuser. Durch den Einsatz von Open Source Software eröffnet sich für eine Stadt- oder Landesverwaltung die Möglichkeit, die lokale Wirtschaft zu stärken, indem lokale Anbieter – natürlich unter Einhaltung der Ausschreibungsrichtlinien – ebenfalls als „Anbieter" und in jedem Fall als „Supportgeber/technische Betreuer" in Frage kommen. Da beim Einsatz von Open Source Software eine Verlagerung der Kosten von den Anschaffungs- bzw. Lizenzkosten hin zu Personalkosten (Implementierung/Adaptierung/Wartung/Support) stattfindet, kommt der Fördereffekt der lokalen Wirtschaft besonders stark zum Tragen.

Bei der Wirtschaftskammer Österreich hat sich in den letzten Jahren eine eigene Expertengruppe[8] formiert, die sich auf die Entwicklung und Implementierung von Open Source Software spezialisiert hat und mittlerweile über 200 Mitglieder aus ganz Österreich verfügt.

2.3.3 Druck auf Anbieter proprietärer Software

Oftmals reicht schon eine Absichtserklärung seitens einer großen Verwaltung, den Umstieg auf Open Source Software intensiv zu überdenken, um die Preisgestaltung von Anbietern proprietärer Software zu „flexibilisieren". In diesem Sinne trägt Open Source Software jedenfalls zur Förderung des Wettbewerbs bei und ist als Bereicherung des Marktes zu verstehen. Auch wenn eine Verwaltung nicht beabsichtigt, sofort oder vollständig auf Open Source Software umzusteigen, sollte diese Alternative dennoch bei Neuausschreibungen oder Preisverhandlungen mit einbezogen werden. Durch die steigende Verbreitung von Open-Source-Produkten kommen Hersteller proprietärer Produkte unter Druck und es kommt zu einer Stärkung der freien Marktwirtschaft. Aus diesem Grund ist Open Source auch ein Thema auf europäischer Ebene und wird der Einsatz solcher Software von der Europäischen Kommission befürwortet bzw. gefördert.

2.3.4 Flexibilität in der Entwicklung und Kostenteilung

Mit dem Einsatz von Open Source Software steigt aufgrund der freien Verfügbarkeit des Quellcodes auch die Flexibilität in der Entwicklung. Anwender sind nicht mehr abhängig von der Entwicklungspolitik eines Herstellers, sondern haben die Möglichkeit individueller Anpassungen. Anbieter technischer Unterstützungsleistungen können mit dem Argument eines Anbieterwechsels ebenfalls leichter unter Druck gesetzt werden. Darüber hinaus eröffnen sich interessante Perspektiven einer verwaltungsübergreifenden Zusammenarbeit durch Austausch und laufende Verbesserung des Quellcodes. Theoretisch wäre aber auch die Entwicklung eines „Kernsystems" im Rahmen einer interkommunalen Kooperation möglich mit individuellen verwaltungsspezifischen Erweiterungen. Damit ließen sich beispielsweise länderspezifische Unterschiede abdecken. Der Kreativität und Flexibilität bei der Entwicklung sind im Open-Source-Bereich jedenfalls kaum Grenzen gesetzt.

Hinzu kommt die – in Zeiten immer knapper werdender Verwaltungsbudgets – besonders attraktive Perspektive

[8] „Open-Source-Experts" der Fachgruppe Unternehmensberatung und IT, im Internet unter www.opensource.co.at zu finden.

- einer Aufteilung der Entwicklungskosten auf mehrere Gebietskörperschaften,
- einer dezentralen Erbringung von IT-Leistungen oder einer
- dezentralen Abwicklung IT-gestützter Aufgaben durch Gebietskörperschaften im Sinne von Kompetenzzentren.

2.3.5 (Verwaltungs-)Kooperationen bei der Softwareerstellung

Im kommunalen Bereich – insbesondere innerhalb eines Bundeslandes – wäre eine Entwicklung gemeindespezifischer Softwaremodule aufgrund des exakt gleichen gesetzlichen Handlungsrahmens durchaus denkbar. Für derartige Entwicklungskooperation bietet Open Source Software den idealen funktionalen wie auch rechtlichen Rahmen, da der Quellcode – wenn er unter die passende Lizenz gestellt wird – beliebig weitergegeben und verändert werden kann. Die Herstellerunabhängigkeit erlaubt eine Entwicklung in jede Richtung, und unter dem Gesichtspunkt einer Kostenteilung ließen sich insbesondere bei Beteiligung einer höheren Anzahl an Gebietskörperschaften zu vergleichsweise niedrigen Kosten für jeden einzelnen Partner maßgeschneiderte Softwareprodukte erstellen. Auch hier gilt, dass aufgrund der Modularität von Open Source Software möglicherweise (oder sogar wahrscheinlich) auf bereits vorhandene Module oder vorhandene Quellcodeteile aufgesetzt werden könnte. Je verwaltungs-unspezifischer der Bedarf ist – wie das beispielsweise bei einer Facility Management Software der Fall wäre – , desto wahrscheinlicher ist es, dass in der Open-Source- Welt bereits Basisprodukte vorhanden sind.

Aufgrund der offenen Lizenzpolitik und der Variabilität bei der eingesetzten Technik (z. B. Web-Services) wäre es im Zuge einer Verwaltungskooperation auch denkbar, dass eine Gebietskörperschaft im Sinne eines Kompetenzzentrums spezifische Verwaltungsaufgaben für die anderen, am Entwicklungsprojekt beteiligten Partnerbehörden übernimmt.

Je gekapselter Open Source Software-Anwendungen von der übrigen Softwarelandschaft der Gebietskörperschaften ablaufen können, desto einfacher wird deren verwaltungsübergreifende Integration. Auch wenn Open Source Software-Entwicklungskooperationen grundsätzlich begünstigt, so darf nicht außer Acht gelassen werden, dass die Gebietskörperschaften – z. B. Städte und Gemeinden – jeweils über eine unterschiedliche, über Jahre hinweg gewachsene Softwarelandschaft – und auch unterschiedliche Ausstattungsniveaus – verfügen. Auch wenn Open-Source-Anwendungen per se multiplattformfähig sind, können sich aus einer Inkompatibilität der in der Verwaltung laufenden Programme (z. B. bei der Datenübergabe oder -übernahme) gegenüber den (offenen) Schnittstellen von Open Source Software-Schwierigkeiten ergeben. Das bedeutet, dass bei einem verwaltungsübergreifenden Open-Source-Projekt im Prinzip alle bestehenden Systeme und Programmversionen aller Partner – insbesondere wenn eine Datenübergabe oder -übernahme stattfinden soll – berücksichtigt werden müssen.

Bei proprietärer Software besteht – abgesehen von einer Mitarbeit in Usergruppen, die vom Hersteller betreut und moderiert werden – kaum eine Möglichkeit, die Entwicklung aktiv mitzugestalten oder mitzubestimmen.

Die Aspekte einer Open-Source-Entwicklungszusammenarbeit in der öffentlichen Verwaltung werden auch auf europäischer Ebene thematisiert. Von der IDA-Arbeitsgruppe[9] der Europäischen Kommission (IDA-Studie) wurde eine umfassende Studienreihe zu diesem Thema ausgearbeitet.

2.3.6 Skalierbarkeit der Systeme

Open Source Software ist grundsätzlich modular aufgebaut, d. h. einzelne Komponenten können entfernt, ausgetauscht oder verändert werden. Es handelt sich nicht um ein Gesamtpaket, sondern um eine individuelle Zusammenstellung von Modulen unterschiedlichen Funktions- und Leistungsumfangs. Dieser Umstand begünstigt die Skalierbarkeit von Systemen, also die Anpassbarkeit an die individuelle Bedürfnislage in einer Organisation.

2.3.7 Kosteneinsparungspotential

Das Thema Kosteneinsparung ist überaus sensibel, da es häufig ungerechtfertigterweise als Hauptargument für den Einsatz von Open Source Software eingebracht wird. Unabhängige Studien haben in der Vergangenheit immer wieder gezeigt, dass es zwar zu einer Verlagerung der Kosten von Anschaffungs- und Lizenzkosten hin zu Personalkosten kommt, das Einsparungspotential aber wesentlich differenzierter zu betrachten ist und sehr stark von Art und Umfang des Open Source Software-Einsatzes abhängt. Es muss berechtigterweise auch die Frage gestellt werden, wie sonstige Stärken und Schwächen von Open Source Software zu bewerten sind.

Es gibt bereits einige Modelle zur Kostenbewertung von Open-Source-Projekten unter Berücksichtigung unterschiedlicher technischer Umsetzungsgrade.

Diese Modelle

- zeigen sowohl kurzfristige wie auch langfristige Kostenfaktoren auf,
- differenzieren zwischen serverseitiger Nutzung von Open Source Software und dem Einsatz auf Arbeitsplätzen,
- berücksichtigen Plattformkosten und Migrationskosten,
- unterscheiden zwischen budgetwirksamen und nicht budgetwirksamen Kosten und
- stellen sowohl kurzfristige wie auch langfristige Kostenfaktoren dar.

Beispiele für verfügbare Kostenmodelle wären die projektorientierte TCO-Berechnung der Gartner Group, das WiBe-Modell der KBST und dessen Anwendung für Open-Source-Projekte, die Modelle der EU (IDA), des dänischen Technologierates oder des holländischen OSOSS-Projekts.

[9] IDA steht für „Interchange of Data between Administrations“ und ist bei der Generaldirektion „Enterprise“ der Europäischen Kommission angesiedelt. Eine Studienreihe von IDA unter dem Titel „Study into the use of Open Source Software in the Public Sector“ beschäftigt sich intensiv mit den europäischen Perspektiven eines Open Source Software-Einsatzes in der öffentlichen Verwaltung. Nähere Informationen zu IDA findet man Internet unter http://europa.eu.int/ida/

Im Zuge einer Kostenbetrachtung wären viele weitere organisationsspezifische und daher individuelle Faktoren zu berücksichtigen wie

- die aktuelle Hard- und Softwareausstattung;
- der kurz-, mittel- und langfristige Erneuerungsplan oder -bedarf;
- die Aktualität der vorhandenen Software bzw. der möglicherweise vorhandene Aktualisierungs- oder Anpassungsbedarf;
- die Rolle des Kostenfaktors in der Gesamtbewertung;
- der Grad und der Bereich der Umstellung (Vollumstellung, Teilumstellung, serverseitige oder clientseitige Umstellung, Mischbetrieb, Umstellung eines/ mehrerer Geschäfts- oder Fachbereiche, Umstellung einzelner Anwendungen u.v.m.);
- Verteilung der Kosten über die Projektlaufzeit usw.

2.4 RISIKEN EINES OSS-EINSATZES

Natürlich ist ein Umstieg auf bzw. Einsatz von Open Source Software mit gewissen Risiken verbunden, die aus den bereits eingangs angeführten Schwächen resultieren. Sind die Risiken bekannt und wurden im Vorfeld eines Open-Source-Projekts in die Machbarkeitsüberlegungen und in die Umsetzungsplanung mit einbezogen, so lassen sich viele zumindest minimieren, wenn nicht ganz ausschließen.

2.4.1 Hoher Umstiegsaufwand und -kosten

Ein Umstieg auf Open Source Software ist natürlich mit erheblichem Aufwand und auch mit Kosten verbunden. Es ist daher besonders wichtig, auch den richtigen Zeitpunkt für ein derartiges Projekt zu wählen und im Vorfeld eine Machbarkeitsstudie sowie eine angemessene Projektplanung durchzuführen. Die Wahl des richtigen Zeitpunkts ist abhängig von einigen verwaltungsinternen Faktoren wie

- Austauschbedarf von Hard- und Software;
- Ablauf von Lizenzen oder Supportverträgen;
- organisatorische Änderungen in der Verwaltung;
- Anforderungen, die von der bisherigen Software nicht erfüllt werden können und kostenaufwendig nachzuimplementieren wären;
- Unzufriedenheit mit bestimmten Softwareprodukten, mit der Stabilität usw.;
- Ablauf der Unterstützung verwendeter Software durch den Hersteller;
- neue, eventuell für die individuellen Einsatzszenarien ungünstige Lizenzmodelle;
- Preissteigerungen durch Hersteller etc.

Der Umstiegsaufwand ist in die Kostenkalkulation mit einzubeziehen und auf die geplante Nutzungsdauer umzulegen. Nicht unberücksichtigt bleiben darf auch der schwer quantifizierbare Aufwand im Bereich des bestehenden Personals, z. B. durch umstellungsbedingte Doppelgleisigkeiten von Systemen, durch längere Bearbeitungszeiten aufgrund mangelnder Vertrautheit mit der neuen Software, Aufwand durch anfängliche Fehlfunktionen oder Bedienungsfehler usw.

2.4.2 Rechtliche Unsicherheit

Die öffentliche Verwaltung ist – zumindest was den behördlichen Bereich betrifft – auf der Grundlage von Gesetzen und Verordnungen tätig. Gerade aus diesem Grund ist es notwendig, dass für die Verwaltung beim Einsatz von Software Rechtssicherheit besteht. Aber auch aus der Perspektive der Kosten und der Investitionssicherung ist es unerlässlich, dass der Einsatz freier Software nicht mit einem Rechtsrisiko verbunden ist. Würde sich herausstellen, dass bestimmte, als Open Source deklarierte oder zertifizierte Software auch nur in Teilen doch nicht frei ist und damit urheberrechtliche Ansprüche entstehen, wäre dies vor allem auf der Kostenseite mehr als unangenehm, auch vor dem Hintergrund, dass eine Umstellung auf Open Source Software nicht einfach rückgängig gemacht werden kann.

Unter diesem Gesichtspunkt sind auch Bedenken der Stadt München im Bereich patentrechtlicher Unsicherheiten und der dringliche Wunsch nach einer Klarstellung zu verstehen. Großes Aufsehen erregte auch ein mittlerweile seit Jahren laufender Urheberrechtsstreit des US-Unternehmens SCO, das für Teile des Linux-Kernels die Urheberschaft reklamiert. Auch wenn viele Rechtsexperten der Auffassung sind, dass der Einsatz von Open Source Software rechtlich unbedenklich ist, bleibt dennoch ein gewisses Restrisiko übrig. Auf europäischer Ebene wird derzeit über eine rechtliche Klarstellung im Bereich Open Source Software diskutiert, insbesondere auch vor dem Hintergrund, dass die Europäische Kommission den Einsatz von Open-Source-Produkten forciert.

Aufgrund der großen Bedeutung der rechtlichen Aspekte eines OSS-Einsatzes für die öffentliche Verwaltung widmet sich ein eigenes Kapitel der vorliegenden Publikation diesem Thema.

2.4.3 Verzögerungen oder Projektstillstand durch unerwartete Probleme

Auch bei bester Projektplanung und -management ist es nicht auszuschließen, dass Probleme bei der Umstellung auftreten. Die Quellen solcher Probleme sind vielfältig und auch bei vorhergehenden Tests nicht vollständig auszuschließen. Mögliche Problemquellen könnten sein

- mangelnde Unterstützung von Spezialhardware,
- Probleme bei der Datenübergabe zwischen Open Source Software und proprietärer Software,
- Integration bestehender Datenbanken bzw. Datenübernahme,
- Inkompatibilitäten innerhalb eingesetzter Softwareprodukte;
- Inkompatibilitäten zwischen Open Source Software und proprietärer Software;
- Funktionsstörungen oder Fehler im Vollbetrieb;
- mangelnde Unterstützung von Funktionalitäten, die aber benötigt werden.

Nicht zu unterschätzen ist die soziale Komponente, auch Widerstand seitens der Bediensteten oder der Personalvertretung kann ein derartiges Projekt – speziell wenn die Umstellung stark auf die Arbeitsplatzsoftware fokussiert – verzögern oder zum Stillstand bringen.

Das Risiko von Verzögerungen lässt sich jedoch durch eine umsichtige und umfassende Projektplanung inklusive Machbarkeitsstudie, vorhergehende Produkt- und Kompatibilitätstest und ein qualifiziertes Projektmanagement reduzieren.

2.4.4 Abhängigkeit von einem Supportgeber

Einer der Vorteile von Open Source Software ist Unabhängigkeit vom Hersteller. Bei jeder Software ist dennoch ein gewisses Maß an Support notwendig. Kaum eine Verwaltung hat jedoch ausreichende Ressourcen, im Falle einer Verwendung von Open Source Software das Know-how für die Pflege der Software verwaltungsintern aufzubauen, vielmehr ist eine Auslagerung des Softwaresupports an externe Dienstleister in Form von Support- und Wartungsverträgen üblich. Damit besteht jedoch die Gefahr eine Abhängigkeit vom Vertragspartner, insbesondere wenn dieser im Auftrag der Verwaltung individuelle Änderungen und Anpassungen der Software vorgenommen hat. Ein Wechsel des Supportpartners ist zwar grundsätzlich möglich, bedeutet aber meist Verzögerungen bei notwendigen Maßnahmen, Einarbeitungszeit durch den neuen Vertragspartner und damit auch Umstiegskosten. Je nachdem, wie sauber und vollständig die Arbeiten des vorhergehenden Supportpartners dokumentiert wurden (z. B. Anpassungen im Quellcode), desto einfacher ist ein Umstieg.

Das Risiko einer Abhängigkeit kann minimiert werden durch eine klare Vertragsgestaltung inklusive einer Berücksichtigung für die Verwaltung günstiger Ausstiegsszenarien und der Auflage für den Vertragspartner, jegliche Quellcodeänderungen und -anpassungen vollständig zu dokumentieren und den Quellcode auch entsprechend zu kommentieren, damit eine Übernahme durch projektfremde Entwickler rasch und ohne hohe Mehrkosten möglich ist.

Ein weiteres Risiko stellen „exklusive Serviceverträge“ dar, die die Verwaltung auf längere Zeit an einen Vertragspartner binden. Solche Verträge kommen insbesondere bei Nischensoftware vor, wenn der Vertragspartner argumentiert, ausschließlich er wäre in der Lage, die Software zu servicieren und Support zu leisten. Exklusive Serviceverträge sind auch bei Open-Source-Produkten zu vermeiden.

2.4.5 Akzeptanzbarriere bei Bediensteten

Auf die soziale Komponente von Open-Source-Projekten wurde bereits bei der Beschreibung möglicher Verzögerungsgründe eingegangen. Tatsächlich darf der Faktor „Mensch“ bei einem derartigen Projekt nicht unterschätzt werden, schließlich betrifft eine Open-Source-Umstellung viele Bedienstete in ihrer höchstpersönlichen Arbeitsumgebung und bedeutet eine von außen angestoßene Veränderung der Arbeitsgewohnheiten. Um dieses Risiko zu minimieren, muss eine offensive Informationspolitik betrieben werden, sollten alle Mitarbeiter von Anfang an – noch vor Projektbeginn, also ab dem Grundsatzentschluss – eingebunden werden und auf deren Ängste eingegangen bzw. diese und auch Informationsdefizite Zug um Zug durch eine Konfrontation mit der Materie abgebaut werden.

Kapitel 3

KRITISCHE ERFOLGS-FAKTOREN UND UMSTIEGSSZENARIEN

Die EDV-Unterstützung hat in der öffentlichen Verwaltung einen hohen Stellenwert eingenommen, ohne entsprechende IT-Systeme wäre eine geregelte Verwaltungstätigkeit undenkbar. Daher gilt die IT schon seit geraumer Zeit als unternehmenskritischer Faktor und ein Ausfall derselben würde den laufenden Geschäftsbetrieb empfindlich stören oder sogar zum Erliegen bringen. Da die IT-Durchdringung auch in der öffentlichen Verwaltung bereits sehr hoch ist, bergen alle Veränderungen oder Umstellungen das Risiko von Fehlfunktionen, Stillständen oder Ausfällen in sich. Eine Umstellung auf Open Source Software – und sei es nur in kleinen, abgegrenzten Teilbereichen – muss daher besonders professionell geplant und durchgeführt werden.

Im Vorfeld eines Austausches proprietärer Software durch Open Source Software sollte man sich daher der kritischen Erfolgsfaktoren bewusst sein und diese entsprechend bei der Planung eines Open Source Software-Umstiegs berücksichtigen – sei es nun flächendeckend über die gesamte Verwaltung, in einem einzelnen Geschäftsbereich oder nur im Bereich eines Softwarepakets.

3.1 KRITISCHE ERFOLGSFAKTOREN VON OSS-PROJEKTEN

3.1.1 Anforderungsanalyse (für Server und Arbeitsplatz) inkl. Auswirkungen

Vor jedem Open-Source-Projekt sollte eine Machbarkeitsstudie stehen, in der die Anforderungsanalyse einen Teilaspekt bildet. Die Gründe für einen Umstieg auf Open Source Software können vielseitig sein, die Bandbreite reicht von politisch-strategischen Faktoren über technische, ökonomische oder qualitative Überlegungen bis hin zu einem Mangel an Alternativen, da am Markt sonst keine passenden „out of the box"-Lösungen[1] verfügbar sind.

Um den Umstellungsaufwand – und damit auch die Kosten – relativ präzise abschätzen zu können, bedarf es einer Aufstellung der umzustellenden Hard- und Software sowie einer Festlegung, ob nach organisatorischen Einheiten (z. B. abteilungsweise) oder nach funktionellen Gesichtspunkten (z. B. alle Webserver) umgestellt werden soll.

Eine abteilungsweise Umstellung bedingt, dass auf einen Schlag mehrere bis viele Anwendungen ausgetauscht werden müssen und damit Risiken wie Inkompatibilitäten, mangelnde Hardwareunterstützung, Funktionsschwächen usw. größer sind. Auf der anderen Seite lassen sich die Probleme auf einen kleinen Benutzerkreis eingrenzen, nämlich die „Testabteilung", und gibt ein gutes Bild, welche Probleme bei einer flächendeckenden Umstellung zu erwarten sind.

Eine Umstellung nach funktionellen Gesichtspunkten ist – vor allem im Serverbereich – wesentlich einfacher durchzuführen, oftmals sogar ohne Veränderungen bei den Arbeitsplätzen der Mitarbeiter, sodass diese vom Softwareaustausch nicht berührt werden und daher auch keinerlei soziale Probleme auftreten. Dafür wirkt eine derartige Umstellung organisatorisch im Regelfall auf große Teile der Verwaltung und kann bei Funktionsdefiziten oder anderen Problemen zu

[1] Unter „out of the box"-Lösungen versteht man fertige Softwareprodukte – Programme, die sofort nach Installation ohne besonderen Anpassungsaufwand verwendet werden können und damit natürlich auch nur über einen beschränkten Funktionsumfang verfügen.

flächendeckenden Störungen des Geschäftsbetriebs führen. Dieses Risiko lässt sich jedoch durch einen parallelen Testbetrieb im Vorfeld weitgehend minimieren. Beispiel für einen flächendeckend wirkenden Austausch im Serverbereich wäre die Ablöse eines bestehenden proprietären Mailservers durch einen Open-Source-Mailserver. Jeder Mitarbeiter mit eigener E-Mail-Adresse wäre von Fehlfunktionen oder einem Ausfall des Mailservers betroffen.

Die Anforderungsanalyse wird idealerweise von der bestehenden IT-Systemlandschaft ausgehen und die aktuellen Anforderungen sowie neue, geplante Aufgaben berücksichtigen. Der Aufwand, die ohnehin abzulösende Soft- und teilweise auch Hardwareausstattung nochmals zu analysieren, macht sich in jedem Fall durch ein geringeres Problemrisiko bei der Open Source Software-Umstellung bezahlt.

Folgende Daten[2] könnten Bestandteil einer Anforderungsanalyse sein:

Informationen zu bestehenden Applikationen (Software)

- Softwarebezeichnung, Versionsnummern (wichtig bei unterschiedlichen Versionen);
- Benötigte Funktionen (im Regelfall wird nur ein Bruchteil des Funktionsvolumens tatsächlich benötigt);
- Anzahl der Anwender, welche die Software benützen;
- Betriebssystem(e), unter dem/denen die Software lauffähig ist (inkl. Terminalserver-Lösungen);
- server- und/oder clientseitige Zusatzsoftware, die für einen reibungslosen Betrieb der Anwendung erforderlich ist (z. B. Middleware, Treiber usw.);
- Minimale Hardwareanforderungen der Software;
- Kommunikationsprotokolle/Schnittstellen zu anderen Systemen;
- Dateiformate, die zum Einsatz kommen;
- Länderspezifische Ausprägungen (Währung, geografische Angaben usw.).

Anforderungen an Daten(transfers)

- Anforderungen an Daten, die in der Verwaltung und an verwaltungsexterne Stellen weitergegeben werden/Datenschnittstellen;
- Notwendigkeit der Datenkonvertierung (von einem System in ein anderes);
- Möglichkeit der Datenkonvertierung (insbesondere bei proprietärer Software ist möglicherweise manchesmal kritisch, bestehende Datenbestände aus einer Applikation herauszubekommen;
- Anforderungen an die Zukunftssicherheit der Datenhaltung.

Sicherheitsanforderungen

- Übernahmemöglichkeit der Benutzer- und Berechtigungsverwaltung einer Software, sofern diese eine eigene benützt;

[2] Die Anforderungsanalyse ist auch wesentlicher Anhaltspunkt für eine Kostenschätzung, für die bereits verschiedene Berechnungsmodelle verfügbar sind. Die angeführten, möglichen Erhebungsdaten basieren auf dem „IDA Open-Source-Migration-Guidelines“, IDA/netproject Ltd., 11/2003

- Anforderungen der bestehenden bzw. der zukünftigen Software an die Benutzeridentifikation (ist beispielsweise eine besondere Hard- oder Softwareausstattung zur eindeutigen Identifikation softwareseitig vorgesehen – z. B. bei softwaregesteuerten Zugangskontrollsystemen);
- Kommen Policies zur Nutzung der Software zur Anwendung und können diese von einer neuen Software erfüllt werden;
- Gibt es spezielle Sicherheitsanforderungen, die nur mit zusätzlicher Hardware erfüllt werden können (z. B. Dongle[3]).

Die Liste möglicher Anforderungen lässt sich beliebig erweitern und sollte in jedem Fall ein exaktes Bild von der individuellen IT-Situation und dem Bedarf in der Verwaltung zeichnen. Die Bedarfsanalyse sollte schlussendlich in einen Migrationsleitfaden münden, aus dem auch der Umstellungszeitplan hervorgeht.

3.1.2 Projektmanagement (systematisches Vorgehen)

Wie bei jedem Projekt ist gutes Projektmanagement der halbe Weg zum Erfolg. Gerade im IT-Bereich kann mangelnde Projektplanung bzw. schlechtes Projektmanagement zu enormen Zeitverzögerungen bis hin zum Projektstillstand sowie zu hohen Mehrkosten durch nicht berücksichtigte und auch nicht budgetierte, individuelle Anpassungsarbeiten führen.

Da Open-Source-Projekten vor allem bei Veränderungen im Client-Bereich, also auf den Arbeitsplatzgeräten der Anwender, garantiert hohe Aufmerksamkeit zuteil wird, ist eine entsprechende Unterstützung durch die Verwaltungsführung unerlässlich. Damit verbunden ist auch die Verpflichtung des Projektleiters, diese laufend über den Stand der Umstellung, über allfällige kritische Phasen oder Probleme zu informieren.

Wie bei jedem Projekt sollte verwaltungsintern eine Steuerungsgruppe zusammengestellt werden, in der der Projektleiter, technische Experten, Vertreter der betroffenen Fachabteilungen oder die Leiter von Fachbereichen, in denen Anwendungen gegen ein Open-Source-Produkt ausgetauscht werden, die Personalvertretung und allenfalls auch externe Experten, die den Prozess begleiten, vertreten sind. Begleitend sind die betroffenen Mitarbeiter intensiv sowie alle übrigen Mitarbeiter in bestimmten Intervallen mit fundierten Informationen zu versorgen, welche Schritte geplant sind, welche Auswirkungen sie haben, welche Ziele verfolgt werden usw.

Im Vorfeld einer Open-Source-Umstellung ist es unerlässlich, sich mit dem Themengebiet intensiv auseinander gesetzt zu haben. Dies betrifft nicht nur die technischen Aspekte, sondern als öffentliche Verwaltung insbesondere auch die rechtlichen Facetten und möglichen Problemzonen eines Open Source Software-Einsatzes.

Die Frage des externen Unterstützungsbedarfs ist zu klären und sollte vor Projektbeginn feststehen. In einer späteren Projektphase zugekaufte Expertenleistungen bringen nicht nur Probleme

[3] Ein „Dongle“ ist ein Stecker, der an eine PC-Schnittstelle angesteckt wird und damit die Verwendung einer bestimmten Software autorisiert. Ohne „Dongle“ würde sich die Software nicht starten lassen bzw. laufen.

aufgrund nicht budgetierter Kosten, sondern auch einen höheren Aufwand, da sich die Experten erst in die bisherigen Umsetzungsmaßnahmen einarbeiten müssen und eventuell bestimmte Teile verworfen werden müssen.

Neben speziellen Anforderungen, die aus der Thematik resultieren, gelten natürlich alle sonstigen Regeln für gutes Projektmanagement bei einem Open-Source-Projekt genau so (Aufstellen eines Projektstrukturplans, einer Ressourcen- und Kostenplanung, einer Definition von zeitlichen und inhaltlichen Zielvorgaben, Meilensteinen usw.).

3.1.3 Externe Beratung (Know-how)

Open Source Software ist ein Spezialgebiet der IT, mit dem IT-Verantwortliche, Systemadministratoren. Experten in der Verwaltung im Tagesgeschäft kaum konfrontiert werden. In größeren Städten, wo Unix[4]-basierte Anwendungen zum Einsatz kommen und auch in der IT-Abteilung höhere Arbeitsteiligkeit bzw. Spezialistentum vorherrschen, werden mehr Know-how im Unix bzw. Open-Source-Bereich und auch mehr Ressourcen für eine intensivere Auseinandersetzung mit neuen Themen angesiedelt sein. Es kann jedoch davon ausgegangen werden, dass – bis auf wenige Ausnahmen – mit der sinkenden Größe der Stadtverwaltung auch das Fachwissen im Open-Source-Bereich abnimmt und bei den IT-Mitarbeitern neben dem Tagesgeschäft kaum freien Ressourcen für eine aufwändige Auseinandersetzung mit Open Source Software vorhanden sind. In der Praxis resultieren gute Open-Source-Kenntnisse bei IT-Verantwortlichen oder Mitarbeitern meist aus privatem Interesse und Engagement.

Externe Beratung verursacht zwar bereits in der Planungsphase eines Open Source Software-Umstiegs-Kosten, sollte dafür aber eine objektive und erfahrene Betrachtung des Projekts, eine qualifizierte Projektplanung und eine weitgehend reibungslose Abwicklung garantieren. Vor allem der externe, unbelastete Fokus auf die gesamte Verwaltung, deren Anforderungen und Probleme, das Einbringen spezifischen Know-hows und die Möglichkeit, vielleicht manchmal auch unangenehme Veränderungen von außen gesteuert einzuführen und auch die Verantwortung dafür zu übernehmen, sind wichtige Erfolgsfaktoren.

Die meisten Städte, die bisher eine Open Source Software-Umstellung in Angriff genommen haben, vertrauten auf externe Expertise, sei es in der Planungsphase als auch bei der Umsetzung.

Die Stadt München, Open-Source-Pionier unter den Großstädten, gab eine Machbarkeitsstudie in Auftrag und wird laufend betreut, die Stadt Wien setzt ebenfalls auf externe Beratung. Schwäbisch Hall, die erste deutsche Stadt, die vollständig auf Open Source Software umgestiegen ist, ging noch einen Schritt weiter und vollzog den Wechsel im Rahmen einer Public Private Partnership (PPP).

[4] Unix ist ein System, auf dessen System- und Anwendungslogik Linux aufsetzt; aus diesem Grund bestehen von der Konzeption und der Handhabung starke Ähnlichkeiten zwischen Unix-Systemen und Open Source Software-Produkten.

3.1.4 Ganzheitliche Kostenbetrachtung

Eine Berechnung der „Total Costs of Ownership“ (TCO) darf gerade bei einem Open Source Software-Projekt nicht fehlen, schließlich sind niedrigere Kosten häufig eines der Hauptargumente für einen Umstieg. Eine nähere Betrachtungsweise relativiert dieses Argument meist relativ rasch und zeigt, dass eher eine Verlagerung weg von Lizenzkosten hin zu Personal- oder Beratungskosten stattfindet. Eine Entscheidung für oder gegen Open Source Software sollte daher nicht ausschließlich von den Kosten abhängig gemacht werden, sondern viel mehr aus einer Abwägung der Vor- und Nachteile aus der individuellen Sicht der Verwaltung erfolgen.

Für die Berechnung der „Total Costs of Ownership“ existieren bereits einige Modelle (z. B. IDA[5]-Berechnungsmodell), die sich bewährt haben. In eine ganzheitliche Kostenbetrachtung gehören die Vorbereitungskosten (z. B. Machbarkeitsstudie, Anforderungsanalyse), die Planungskosten, die Implementierungskosten, die Kosten für Open Source Software[6], Kosten für auszutauschende oder aufzurüstende Hardware, Kosten für zusätzlich benötigte proprietäre Software (z. B. Terminal Server), Kosten für Schulungsmaßnahmen, externe Beratung u. v. m. Eine realistische Kostenbetrachtung ist ebenso wie eine qualifizierte Projektplanung einer der wesentlichsten kritischen Erfolgsfaktoren. Fällt die TCO-Berechnung zu hoch aus, besteht die berechtigte Gefahr, dass das Umstellungsprojekt bereits in der Vorbereitungsphase abgebrochen wird. Sind die kalkulierten Kosten zu niedrig, sind budgetäre Engpässe die Folge, die entweder Abstriche bei den geplanten Maßnahmen zur Folge haben, zu einer schlechteren Umsetzungsqualität führen oder aber eine Verlangsamung des Projekts bis hin zu einer Sistierung nach sich ziehen.

3.1.5 Zeitpunkt und Zeitrahmen für Umstieg

Ein kritischer Erfolgsfaktor ist sicherlich auch der richtige Zeitpunkt eines Umstiegs. Dieser wird aber vollinhaltlich erst nach Abschluss der Anforderungsanalyse und einer Erhebung der umzustellenden Hard- und Software festzustellen sein. Vorab-Indikatoren sind aber

- Auslaufen der Lizenzen bei größeren Softwarepaketen bzw. bei höheren Stückzahlen;
- Preiserhöhungen für Lizenzen proprietärer Software;
- Ablösebedarf im Bereich der Hardware;
- Neue Software wird benötigt, die in der Verwaltung noch nicht vorhanden ist;
- Mit bestehender Software wird nicht mehr das Auslangen gefunden;
- Für benötigte Software gibt es keine brauchbaren kommerziellen Produkte usw.

Gerade in der öffentlichen Verwaltung gibt es oftmals Vorbehalte, einen Trend frühzeitig aufzugreifen. Bei Open Source Software kann mittlerweile aber nicht mehr von einer frühen Phase gesprochen werden und einige „early adopters“ haben bereits vorgezeigt, wie ein Umstieg ablaufen kann, welche Probleme auftreten und welche Fehler passieren können. Es kann also auf umfassende Erfahrungswerte zurückgegriffen werden.

---

[5] IDA = Interchange of Data between Administrations

[6] Auch bei Open Source Software können Kosten anfallen; Open Source bedeutet nicht zwangsläufig, dass die Software gratis ist.

Der Zeitrahmen für einen Umstieg ist natürlich abhängig vom Umfang des Projekts, generell sollte der zeitliche Rahmen aber möglichst knapp gehalten und die Umstellung rasch vollzogen werden. Hauptargumente dafür sind die außerordentlichen Belastungen der IT-Abteilung während eines derartigen Projekts, die Gefahr, dass sich das Projekt über einen längeren Zeitraum hinweg totläuft, und eine unnötige emotionale Mehrbelastung der Mitarbeiter, für die eine Open Source Software-Umstellung möglicherweise einen Eingriff in ihr persönliches Arbeitsumfeld bedeutet.

3.1.6 Sicherung eines professionellen Supports

Professioneller Support ist das Um und Auf jedes techniklastigen Projekts. Bereits im Vorfeld der Umstellung sollten geeignete technische Partner gefunden werden, die das Projekt auch von Anfang an begleiten. Nur so kann sichergestellt werden, dass die neu geschaffene oder in Teilbereichen ausgetauschte Systemlandschaft dem Partner im Detail bekannt ist und eine effiziente und effektive Nachbetreuung möglich ist. Eine Variante, einen Supportgeber an die Verwaltung zu binden, wäre der Abschluss einer Public Private Partnership.

Umgekehrt besteht die Gefahr einer zu engen Bindung an einen speziellen Supportgeber, insbesondere wenn dieser an der Softwareerstellung oder -modifikation mitgewirkt hat. In diesem Fall entsteht leicht ein starkes Abhängigkeitsverhältnis zum Supportgeber – ein Umstand, der gerade bei Open Source Software kontraproduktiv ist und dem Grundgedanken von Open Source entgegenläuft.

Der große Vorteil von Open Source Software – das Vorliegen des veränderbaren Quellcodes – bleibt in jedem Fall erhalten, ein Wechsel des Supportgebers ist – im Gegensatz zu mancher proprietärer Software – immer möglich. Je nach Qualität der Dokumentation des Quellcodes bzw. der Softwarefunktionalitäten entstehen durch einen Wechsel aber in jedem Fall Kosten.

3.1.7 Dokumentation

Eine umfassende Dokumentation ist in vielerlei Hinsicht ein kritischer Erfolgsfaktor, auch wenn die Auswirkungen häufig erst später – nach erfolgter Umstellung – zum Tragen kommen. Wird Software auf Open-Source-Basis neu erstellt, so ist eine Dokumentation des Quellcodes unerlässlich, damit sich andere Programmierer rascher (oder überhaupt) zurechtfinden und die Abläufe im Code verstehen. Mit einer guten Quellcode-Dokumentation wird ein Wechsel der/des technischen Partner/s in jedem Fall einfacher.

Aber auch die Dokumentation des Funktionsumfangs und der Anwendung von Open Source Software hat Bedeutung. Schließlich gibt es bei Open Source Software – noch dazu, wenn sie verändert wurde – keine „vom Hersteller autorisierten“ Handbücher, Leitfäden usw. Umso wichtiger ist eine gute individuelle Dokumentation, die auch den bestehenden Mitarbeitern und neu aufgenommenen Mitarbeitern ein leichteres und vor allem rascheres Einlernen in neue Software erlaubt.

3.1.8 Integration/Beteiligung der Mitarbeiter, Schulungen

Der wahrscheinlich kritischste Erfolgsfaktor sind die Mitarbeiter. Handelt es sich um eine serverseitige Open Source Software-Umstellung, so merken die Mitarbeiter unter Umständen nichts davon. Die meisten Umstellungen betreffen aber zumindest Teilbereiche der Clients, also der EDV-Arbeitsplätze.

In dem Moment, in dem es zu Veränderungen auf den Arbeitsplatz-PCs von Mitarbeitern kommt, werden auch Widerstände und Gegenargumente nicht lange auf sich warten lassen. Der Beharrungsmoment ist bei EDV, insbesondere bei älteren Bediensteten, sehr hoch. Vielfach wird IT-Ausstattung (z. B. ein Notebook, ein Flachbildschirm u.Ä.) auch heute noch als Prestigefaktor betrachtet. Eine Umstellung auf Open Source Software im Clientbereich bedeutet – je nach Umstellungsgrad – für Mitarbeiter einen Eingriff in deren täglichen Arbeitsablauf und berührt diese unmittelbar und persönlich. Eine entsprechende Einbindung der Mitarbeiter ist daher oberste Regel bei derartigen Projekten.

Bereits im Vorfeld, also noch vor Anlaufen der Projektplanung, sollte eine Aufklärungskampagne unter den betroffenen Mitarbeitern begonnen werden. Wichtig wird es sein, ein Bewusstsein für die Bedeutung des Projekts zu schaffen. Das kann aber nur gelingen, wenn jeder grundsätzlich weiß, was mit Open Source Software gemeint ist, wodurch sie sich auszeichnet und wie sie sich von herkömmlicher Software unterscheidet. Wenn es bereits im Vorfeld gelingt, durch gezielte Informations- und Aufklärungspolitik den Mitarbeitern die Angst vor den geplanten Veränderungen zu nehmen und diese ins rechte Licht zu rücken, dann wird auch die Akzeptanz und Mitarbeit entsprechend besser sein.

Bei den meisten Projekten – und zwar nicht nur im IT-Bereich – sind die Mitarbeiter der kritische Erfolgsfaktor schlechthin, schließlich sind sie es, die die Veränderungen annehmen und damit leben müssen. Diesem Umstand sollte angemessen Rechnung getragen werden.

3.2 UMSTIEGSZENARIEN AUF OPEN SOURCE SOFTWARE

Es gibt einige Gründe für die öffentliche Verwaltung, die dafür sprechen, auf Open Source Software umzusteigen.[7] Vor dem Umstieg auf Open Source-Programme ist es jedoch wichtig, dass die tatsächlichen Gründe für eine Migration klar sind, diese von den IT-Mitarbeitern und Usern aktiv unterstützt wird und sichergestellt ist, dass jeder Schritt des Umsetzungsprozesses auch durchführbar ist.[8]

Beim Austausch von IT-Systemen muss beachtet werden, wie die Interoperabilität[9] von Systemen sichergestellt werden kann, wie mobile User unterstützt werden können, wie steuerbare Systeme aufgebaut werden können u. v. m. Wird proprietäre Software ausgetauscht, muss überprüft werden,

[7] Siehe Kapitel 2.
[8] Vgl. netproject, 2003, S. 12.
[9] Interoperabilität meint die Fähigkeit, von verschiedenen Programmen die selben Dateiformate zu schreiben und zu lesen und die selben Protokolle zu verwenden.

ob die Templates[10] den korrekten Output produzieren und Makros[11] müssen überschrieben werden. Applikationen, für die es keine äquivalente Open Source Software gibt, können als „thin clients“[12] der Terminalservices geführt werden.

Obwohl ein kompletter Umstieg auf Open Source Software am sinnvollsten wäre, sind Mischformen von Open-Source- und proprietären Applikationen ebenso Praxis, denn ein gesamter Austausch der Programme durch Open-Source-Applikationen ist insbesondere bei der Vielzahl an verwaltungsspezifischen Fachapplikationen nicht immer möglich oder vorteilhaft.[13]

Jedes Migrationsprojekt sollte generell aus folgenden Phasen bestehen[14]:

- Datensammlungs- und Projektdefinitionsphase.
- Erstellen einer Rechtfertigung für den Umstieg und Aufstellung der damit verbundenen Kosten (z. B. TCO[15]-Berechnung, Projektkosten etc.).
- Eine oder mehrere Projektphasen, die darauf ausgerichtet sind, den Projektplan und die Rechtfertigung für die Durchführung zu testen.
- Durchführung des Projekts.
- Durchführung eines Soll-Ist-Vergleichs.

3.2.1 Durchführung des Projekts

Ein Umstieg zu Open Source Software gestaltet sich in jeder Organisation individuell, idealerweise sollte sich ein Migrationsprozess jedoch aus folgenden Teilen zusammensetzen[16]:

1. Zusammenstellung eines Teams mit qualifizierten Mitgliedern, die ausreichend Unterstützung seitens der Leitung bekommen.
2. Sensibilisierung der Mitarbeiter für die Ziele des Projekts und der Aspekte von offener Software, teilweise mit externer Unterstützung durch Berater.
3. Sicherstellung, dass Aspekte wie Bedeutung von Open-Source-Lizenzen, Vor- und Nachteile verschiedener Produkte oder Unterschiede zwischen einzelnen Distributionen klar sind.
4. Auditierung von bestehenden Open-Source-Systemen. Es soll hierbei herausgefunden werden, welche Aspekte im Zusammenhang mit Applikationen berücksichtigt werden müssen (benötigtes Betriebssystem, Hardware, Dateiformate etc.), welche Datenanforderungen bestehen (z. B. Beschränkungen an Schnittstellen zu Systemen) und welche Anforderungen an die Sicherheit gestellt werden (z. B. Benutzername- und Passwort-Verwaltung, Einschränkungen beim Internetgebrauch).

---

[10] Templates ermöglichen das Schreiben des Codes ohne Berücksichtigung des Datentyps, mit dem er möglicherweise verwendet wird.

[11] Makros ersetzen ein bestimmtes Textmuster entsprechend eines definierten Satzes von Regeln. Makros dienen zur Automatisierung von häufig genutzten Sequenzen oder zur Ermöglichung einer stärkeren Abstraktion.

[12] Die Anwendungen laufen auf einem zentralen Gerät mit der ursprünglichen Softwareausstattung, allerdings wird darauf von dezentralen Clientcomputern über ein Netzwerk zugegriffen.

[13] Vgl. netproject, 2003, S. 12.

[14] Vgl. ebenda, S. 14.

[15] Total Costs of Ownership.

[16] Vgl. netproject, 2003, S. 17 ff.

5. Erstellung eines Business-Plans basierend auf den zuvor gesammelten Daten.
6. Darstellung der Gründe für eine Migration gegenüber den Usern und Entwicklung eines Helpdesks für die Anliegen der User und zur Problembehebung.
7. Durchführung von Pilotprojekten mit einer kleinen Anzahl von Usern.
8. Festlegung der Geschwindigkeit des Umstiegs. Entweder wechseln alle User vom alten ins neue System am selben Tag („Big Bang Methode“) oder die Umstellung erfolgt phasenweise pro Gruppe bzw. pro User. Grundsätzlich ist es üblich, dass alte und neue Systeme über einen Zeitraum parallel betrieben werden.
9. Migration der gesamten Verwaltung.
10. Einholung des Feedbacks der User und Behandlung von auftretenden Problemen.

Um die Einführung von Open-Source-Programmen zu erleichtern, empfiehlt es sich, neue Applikationen zunächst in bekannte Umgebungen zu integrieren: viele Open-Source-Applikationen arbeiten auch auf proprietären Betriebssystemen und können daher auch ohne den kompletten Austausch der Umgebung eingeführt werden (z. B. arbeiten OpenOffice, Mozilla und Apache unter Windows und können als Ersatz für MS Office, Internet Explorer und Internet Information Server eingesetzt werden). Ebenso sollen Veränderungen zuerst auf den Servern durchgeführt werden, da dies weitgehend unabhängig von den Usern und ohne diese zu beeinträchtigen geschieht. Dadurch wird auch eine Plattform für die spätere Einführung von Veränderungen an den Clients geschaffen. Darüber hinaus ist zu beachten, dass nichts durchgeführt wird, was einen zukünftigen Umstieg erschweren könnte, z. B. ist darauf zu achten, dass alle Web-Entwicklungen einen Inhalt produzieren, der auf allen gängigen Web-Browsern, speziell auf Open-Source-Browsern betrachtet werden kann.[17]

3.2.2 Beispiel einer Migration von Windows 2000 auf Open Source Software

Der mögliche Ablauf einer Migration wird im Folgenden kurz anhand der deutschen Monopolkommission skizziert.

Entstehung des Projekts
Das deutsche Innenministerium ergriff 2002 die Initiative zur Einführung von Open-Source-Projekten in Bundesbehörden, um die Einsatzfähigkeit von solchen Programmen und speziell von Linux zu testen und zu bewerten. Die Monopolkommission erklärte sich daraufhin bereit, ein Pilotprojekt zu initiieren.[18]

Vorbereitung des Projekts – Konzepterstellung
Das Ziel der Server-Migration war eine flexibel einsetz- und erweiterbare Client/Server-Landschaft, weswegen alle Server-Komponenten an die Anforderungen der Linux-Clients angepasst wurden. Ziel der Client-Migration war der Austausch der bestehenden Applikationen durch adäquate Open-Source-Applikationen oder Applikationen, die unter dem Betriebssystem GNU/Linux laufen können. Beinahe alle Applikationen, die auf dem Betriebssystem Windows 2000 liefen,

[17] Vgl. netproject, 2003, S. 23 f.
[18] Vgl. Sprickmann Kerkerinck, 2004, S. 29.

wurden durch freie Software oder Open-Source-lauffähige Applikationen ersetzt. Als Office-Applikation entschied man sich für Staroffice 6.0[19], der Windows-Explorer wurde durch GNOME Midnight Commander ersetzt, welcher, basierend auf Linux, die selben Funktionen bietet. Für die Mail-Funktionalität wurde Sylpheed, als Browser Galeon und als Scanner-Software SANE ausgewählt.[20]

Für die Anmeldung am Client wurde eine Kombination von zwei Sicherheitsmerkmalen zur Authentifizierung angedacht: die Kombination von Chipkarte und Fingerabdruck. Für die Herstellung der Chipkarten und die Erfassung der Fingerabdrücke wurde eine spezielle Applikation entwickelt. Mit der Chipkarte kann sich der Anwender sowohl online als auch offline anmelden, d. h. auch ein Laptop, der sich nicht immer im Netzwerk befindet, kann als Client eingesetzt werden und wird zentral mit allen Applikationen ausgestattet, sobald er sich im Netz befindet. Im Rahmen der Migration sollte auch eine digitale Signatur für die Gewährleistung eines gesicherten Mailverkehrs eingeführt werden. Dies wurde in Form von GNUpg[21]-Schlüsseln realisiert, wobei GNUpg-Daten verschlüsselt auf der Chipkarte abgelegt werden. [22]

Neben den Lösungen für Client und Server sollten im Zuge der Migration noch weitere Funktionalitäten erstellt oder ersetzt werden. Dazu zählen u. a. eine Bibliotheksverwaltung, CD-Archivierung, Kalender, Adressbuch, Migration Fragebogen-Workflow und eine Statistikdatenauswertung.[23]

Aufbau der Testumgebung
Innerhalb der ersten vier Wochen wurden einzelne Mitarbeiter mit Linux-Arbeitsplätzen ausgestattet, wodurch diese die Applikationen bereits zu einem frühen Zeitpunkt testen und Anregungen geben konnten. Die hier eingesetzten Applikationen wurden zuvor bereits auf ihre Eignung getestet und ausgewählt. Als wesentlichste Umstellung für die Mitarbeiter erwies sich die Anwendung der Sicherheitsmechanismen (Chipkarte und Fingerabdruck).[24]

Abwicklung der Migration
Aufgrund der aufwendigen Vorarbeiten erfolgte die tatsächliche Umstellung der Clients und der Serverumgebung erst kurz vor Projektende. Die Migration der Serverumgebung erfolgte durch die Aufstellung des entsprechenden File- und Webservers in der Testumgebung, der bisher eingesetzte Fileserver wurde parallel dazu im Netz betrieben. Den Mitarbeitern wurden Fingerabdrücke entnommen und erklärt, wie die Anmeldung mittels Chipkarte und Fingerabdruck funktioniert. Erst nach der vollständigen Migration der Clients wurde der Fileserver umgestellt. Da es dafür keine Testmöglichkeiten gab, erfolgte die Umstellung am Wochenende. Bei einem eventuellen Auftreten von Problemen wäre es möglich gewesen, auf die vorhandene Fileserver-Umgebung in der Testumgebung zurückzugreifen.[25]

[19] Basiert auf dem OS-Produkt OpenOffice, ist aber selbst nicht OSS, da lizenzpflichtige Produkte hinzugefügt wurden.
[20] Vgl. Spickmann Kerkerinck, 2004, S. 31 f.
[21] GNU Privacy Guard (GNUpg) wird für digitale Verschlüsselung und digitale Signatur verwendet und ist auch für eine offline-Kommunikation (E-Mail) und Speicherung von Daten einsetzbar.
[22] Vgl. Spickmann Kerkerinck, 2003, S. 34 f.
[23] Vgl. ebenda, S. 37 ff.
[24] Vgl. ebenda, S. 39 f.
[25] Vgl. ebenda.

Schulung der Benutzer und Support

Nach der Migration der Clients wurden spezielle Benutzerschulungen für die verschiedenen Aufgabenfelder der Mitarbeiter durchgeführt. Die Mitarbeiter erhielten eine individuelle Einschulung, die darauf abgestimmt war, nebenbei ausreichend Zeit für das Tagesgeschäft zu lassen und den Mitarbeitern die Möglichkeit bot, bei der Auswahl der benötigten Module mitzubestimmen. Nachdem die Arbeit im Produktivsystem begonnen hatte, wurde eine Zeitlang ein Supportservice in Form von Hotline- und E-Mail-Support angeboten. Im Rahmen des Supports wurden Aspekte aufgedeckt, die während der Projektdurchführung aufgrund des knapp bemessenen Zeitrahmens (drei Monate) nicht ausreichend berücksichtigt werden konnten.[26]

Weitere Schritte

Die nächsten Umsetzungsschritte sind die Implementierung einer VPN-Lösung[27] und ein Update der Office-Umgebung oder die Implementierung der gewünschten Funktionen aus anderen Applikationen.[28]

Literaturverzeichnis

netproject Ltd.: The IDA Open-Source-Migration-Guidelines, 2003.

Spickmann Kerkerinck, Thomas: Beispiel einer Migration von Windows 2000 auf Open Source Software, in: Gehring, Robert A. / Lutterbeck, Bernd (Hrsg.): Open-Source-Jahrbuch 2004/Zwischen Softwareentwicklung und Gesellschaftsmodell, Berlin, 2004, S .29-43.

[26] Vgl. Sprickmann Kerkerinck, 2003, S. 40 f.

[27] Virtual Private Network: Mitarbeiter können damit über eine sichere Verbindung auch von außen über das Internet auf ihre Daten zugreifen.

[28] Vgl. Spickmann Kerkerinck, 2003, S. 41.

Kapitel 4

WIRTSCHAFTLICHE ASPEKTE VON OPEN SOURCE SOFTWARE IN DER ÖFFENTLICHEN VERWALTUNG

Die öffentliche Verwaltung ist bei ihren Entscheidungen über den Einsatz unterschiedlicher Alternativen im Bereich Software mit zwei Dimensionen des Problems konfrontiert: Einerseits geht es im Sinne der Sparsamkeit der Verwaltung darum, möglichst effiziente Lösungen mit einem betriebswirtschaftlich optimalen Kosten-Nutzen-Verhältnis zu wählen, andererseits sind auch die wirtschaftlichen Folgeeffekte für die Unternehmen und für die Bevölkerung in der Region von besonders großer Bedeutung. Die Diskussion über die Investitionen in Software allgemein und um den Einsatz von Open Source Software bzw. proprietären Lösungen im öffentlichen Sektor im Speziellen bezieht sich daher neben den Kosten (TCO – total cost of ownership) auch auf die gesamtwirtschaftlichen Effekte unterschiedlicher Arten von Software.

Bei den betriebswirtschaftlichen Faktoren kommt es durch eine Einführung von Open Source Software vor allem zu einer Verschiebung weg von den Anschaffungskosten hin zu Entwicklungs-, Betriebs- und Supportkosten. Es zeigt sich dabei, dass Open Source Software entgegen der verbreiteten Auffassung ebenfalls meist kommerziellen gewinnorientierten Geschäftsmodellen folgt. Daher existieren ökonomische Anreize für die Entwicklung von Open Source Software. Auf der anderen Seite stellt sich heraus, dass die gesamtwirtschaftlichen Effekte von Software nur zum Teil vom Entwicklungsmodell oder Geschäftsmodell abhängen, sondern maßgeblich von anderen Faktoren (wie z. B. Grad der Standardisierung, Verbreitung oder Wiederverwendbarkeit) bestimmt werden. Die gesamtwirtschaftlichen und regionalökonomischen Effekte von Open Source Software stellen sich in der Regel etwas geringer dar als jene von proprietärer Software. Die empirischen Ergebnisse legen nahe, dass der öffentliche Sektor im Rahmen von Ausschreibungen auf Interoperabilität und die Einhaltung von Standards achten sollte, während eine darüber hinausgehende aktive Förderung bestimmter Arten von Software aus ökonomischer Sicht nicht sinnvoll oder erforderlich ist.

4.1 WIRTSCHAFTLICHE DIMENSIONEN DER SOFTWAREENTSCHEIDUNGEN IM ÖFFENTLICHEN SEKTOR

Generell befindet sich die öffentliche Hand in der Situation, Entscheidungen über die Art und Strukturierung ihrer Leistungserstellung („Produktionsprozesse“) anhand mehrerer Dimensionen zu beurteilen:

- Kostendimension
- Nutzendimension
- Folgeeffekte durch die Nachfrage nach bestimmten Produktionsfaktoren

Während die ersten beiden Dimensionen jedes wirtschaftliche Unternehmen ebenfalls betreffen, stellt die Bedeutung der letzten Frage ein Charakteristikum des öffentlichen Sektors dar, da das Volumen und die Signalwirkung der Entscheidungen der öffentlichen Hand in einigen Fällen dazu führen können, dass Marktprozesse beeinflusst werden.

Bei der Erstellung öffentlicher Leistungen tritt jedoch noch ein weiteres Problem auf: sehr oft kann der Wert dieser Leistungen nicht bestimmt werden, da die „Produkte“ (Verwaltungsleistungen, …) des öffentlichen Sektors meist nicht am Markt „verkauft“ werden und daher statt der üblichen Bewertung mit Marktpreisen alternative Bewertungsverfahren zur Anwendung gelangen müssen.

In der Regel wird der Wert dieser Leistungen daher in der Praxis mit den Herstellungskosten gleichgesetzt. Daher sind Kosten und Nutzen der öffentlichen Leistungen in der Kosten-Nutzen-Analyse definitionsgemäß gleich, sodass das Entscheidungskriterium Kosten-Nutzen-Relation „mangels vorhandener Relation“ nicht als Entscheidungsgrundlage herangezogen werden kann. Aus diesem Grund werden daher üblicherweise lediglich die Kosten betrachtet (zentrales Konzept, insbesondere im IT-Bereich: TCO – total cost of ownership). Unterstellt wird dabei, dass bei gegebenem Output-Niveau und gegebener Output-Qualität der Verwaltungsprozesse (oder auch der IT-Prozesse in einem Unternehmen) die Minimierung der gesamten Kosten des betroffenen Bereiches auch ein betriebliches Optimum darstellt.

Für eine entscheidungsorientierte Betrachtung ist dies jedoch kein gangbarer Weg, da die unterschiedlichen Alternativen verschiedene Vorteile und Nachteile aufweisen, die einen Vergleich lediglich auf der Basis der Kosten nicht zulassen (das berühmte „Vergleichen von Äpfeln und Birnen“). Es müssen also auch die Nutzeffekte ausreichend bestimmt werden. Dies kann auf verschiedene Arten erfolgen:

- Normierung eines Anforderungsprofils und Kostenvergleich von Alternativen, die diese Kriterien erfüllen. Es stellt sich dabei die Frage, wie an sich nicht erwartete Zusatzfunktionalitäten wertmäßig berücksichtigt werden sollen. Eine strenge Auslegung der wirtschaftswissenschaftlichen Lehrmeinung würde diesen Features gar keinen Wert beimessen.

- Einbeziehung aller Nutzeffekte, die direkt oder indirekt durch die Alternativen zu erwarten sind. Die Bewertung dieser Nutzeffekte ist jedoch äußerst problematisch.

Letztendlich wird man aufgrund dieser methodischen Probleme in der Praxis auf betriebswirtschaftlicher Ebene wieder bei einer (modifizierten) Betrachtung der TCO ankommen,

Die Analyse der wirtschaftlichen Folgeeffekte durch den Einsatz einer bestimmten Technologie kann nur anhand (regional)ökonomischer Modelle erfolgen und stellt das zweite Standbein für eine fundierte Technologiewahlentscheidung des öffentlichen Sektors dar. Auch hier hat wieder eine Abwägung betriebs- und volkswirtschaftlicher Effekte zu erfolgen.

4.2 CHARAKTERISTIKA VON SOFTWARE

4.2.1 Wirtschaftliche Eigenheiten des Produktionsfaktors „Software“

Die ökonomische Theorie basiert in der Regel darauf, dass der Marktmechanismus auf funktionsfähigen Märkten insbesondere über den Preismechanismus ausreichende Informations-, Anreiz- und Steuerungsfunktion ausübt, sodass sich in der Folge ein „effizientes Marktgleichgewicht“ einstellt. Das Gut „Software“ weist jedoch aufgrund seiner besonderen Eigenschaften bestimmte Spezifika auf, die von der ökonomischen Standardtheorie deutlich abweichen:

Da es sich bei Software um ein Gut an der Grenze zwischen Sachgut, Dienstleistung und Wissen handelt, kommt der leichten Kopierbarkeit große Bedeutung zu. Das für „normale“ Märkte geltende

Prinzip der „Ausschließbarkeit anderer Personen“ von der Nutzung des Gutes kann nur eingeschränkt wirken, ebenso herrscht wegen der einfachen Kopierbarkeit eine nur eingeschränkte „Rivalität im Konsum“ zwischen den einzelnen Personen. Diese beiden Mängel werden auch als Charakteristik eines **öffentlichen Gutes** (im Sinne der Wirtschaftstheorie) bezeichnet. Da Software ein technologisch hochwertiges Gut ist, das auf intensiver Forschung und Entwicklung beruht, kommt es dadurch für den einzelnen Anbieter auch zu geringeren Innovationsanreizen. Es entstehen somit starke (positive) „externe Effekte“ bei der Entwicklung von Software, die dem Entwickler nicht oder nur unzureichend abgegolten werden, die also keinen betriebswirtschaftlichen Anreiz bilden können.

Normalerweise ist ein Individuum indifferent, ob jemand anderer das gleiche Gut konsumiert oder nicht. Für Software trifft diese „Gleichgültigkeit“ aber nicht zu, da der Nutzer von Software große Vorteile davon haben kann, da das von ihm verwendete Programm auch von anderen Personen genutzt wird (Kompatibilität, höhere Qualität durch bessere Praxiserprobung, Interoperabilität, Vermeidung von Schnittstellenproblemen, bessere Nutzbarkeit des Anwendungswissens an anderen Arbeitsplätzen, ...). Diese Zusammenhänge werden auch als **Netzwerkeffekte** bezeichnet.

Zu den Netzwerkeffekten kommt noch hinzu, dass Software durch extrem hohe Vorlaufkosten und äußerst niedrige Grenzkosten (zusätzliche Kosten für den Verkauf einer weiteren Einheit des Produkts) gekennzeichnet ist. Die ökonomische Theorie spricht in solchen Fällen von einem **natürlichen Monopol**, da ein einziger großer Anbieter mit nur einem Produkt den Markt aufgrund starker Fixkostendegression (und geringer Grenzkosten) zu geringeren Kosten versorgen kann, als dies der Fall wäre, wenn mehrere Anbieter sich den Markt teilen würden und damit jeder einzelne Anbieter seine hohen Entwicklungskosten auf eine geringere Anzahl von verkauften Produkten aufteilen müsste. Dies gilt für alle Arten von Software gleichermaßen – der Entwickler hat schon aufgrund des Wissensvorsprunges eine bestimmte „Monopolmacht“, da andere Anbieter wegen der erforderlichen Einarbeitung in das Produkt mit sehr hohen Anlaufkosten konfrontiert wären, die sie in die Kundenpreise mit einrechnen müssten und somit teurer sein müssten als der ursprüngliche Hersteller.

Die genannten Marktunvollkommenheiten führen nach der unbestrittenen Auffassung der Volkswirtschaftstheorie zu dem Problem des „partiellen Marktversagens“. Der Softwaremarkt kann also nicht von selbst effizient die Aufgaben eines Marktes erfüllen, ohne dass eine Intervention von außen erfolgt.

4.2.2 „Marktversagen“ auf dem Softwaremarkt

Im ökonomischen Normalfall wird auf dem Markt bestimmt, wer wie viel eines knappen Gutes erhält (Distributionsfunktion des Marktes), indem über den Marktpreis und die Zahlungsbereitschaft des Einzelnen eine Zuteilung der Güter in dezentraler Weise erfolgt. Dieses Grundprinzip der Marktwirtschaft wird jedoch bei Software durchbrochen.

Einerseits kann Software (ohne komplexe Kopierschutzmechanismen) in der Regel leicht vervielfältigt werden, auf der anderen Seite wird der Nutzer von Software meist nicht dadurch eingeschränkt, dass ein anderer Nutzer **dieselbe** Software verwendet. Dazu bietet sich folgende

Analogie an: Der Käufer z. B. einer Wurstsemmel wird diese in der Regel nicht unentgeltlich oder unter ihrem Wert an Dritte weitergeben, da er dadurch selbst auf den Konsum dieser speziellen Wurstsemmel verzichten muss. Bei Software wird diese Einschränkung nicht wirksam, da das „digitale Produkt" ohne Substanzverlust und mit sehr geringen Kosten quasi „geklont" werden kann. Oft hat der Weitergebende sogar im Falle einer Gratisabgabe noch einen zusätzlichen Vorteil von der Weitergabe, da er entweder von der Austauschbarkeit von Anwendungsdaten (Datendateien, Dokumente, Tabellen etc.) profitiert oder damit rechnen kann, dass er im Gegenzug ebenfalls andere Software gratis zur Verfügung gestellt bekommen wird.

Ohne weitere Interventionen kann diese Unvollkommenheit schließlich dazu führen, dass bestimmte Arten von Software gar nicht mehr entwickelt werden, weil für Unternehmen keine Gewinnmöglichkeiten bestehen und somit auch kein Markt entstehen kann (analog zum klassischen Lehrbuchbeispiel des Leuchtturms, den niemand freiwillig bauen würde, da derjenige allein die Kosten tragen müsste, die gesamte Gesellschaft aber davon profitieren würde). Aber auch in gemäßigten Situationen, in denen es zu keinem vollkommenen Zusammenbruch des Marktes kommt, entsteht eine Unterversorgung mit dem von dieser Art von Marktversagen behafteten Gut.

Die in der Ökonomie allgemein vorgeschlagenen und in der Realität anzutreffenden Lösungsvorschläge beruhen auf voneinander vollkommen verschiedenen Konzepten:

- Einerseits könnte der Staat die Produktion des Gutes als öffentliche Aufgabe übernehmen und die Entwicklungskosten zwangsweise von seinen Bürgern z. B. über Steuern oder Abgaben einheben (öffentliche Bereitstellung; Verstaatlichung). Diese Lösung hat den Nachteil, dass durch den Ausschluss von Wettbewerb anstelle der Präferenzen der einzelnen Nachfrager die zentralen Entscheidungen einer Planungsinstanz bestimmen, welche Software mit welchen Leistungsmerkmalen entwickelt und zur Verfügung gestellt werden soll, und ist in der Praxis heutzutage sehr umstritten (da es bei der öffentlichen Bereitstellung im Gegenzug zum Problem des „Staatsversagens" kommen kann – mangelhafte Effizienz etc.).

- Die marktkonformere Lösung zielt darauf ab, die Unvollkommenheiten selbst zu mildern oder zu beheben: Da das Problem hauptsächlich in der mangelnden Ausschließbarkeit und der mangelnden Rivalität im Konsum begründet ist, kann der Staat durch den Schutz des geistigen Eigentums und das Verbot der unberechtigten Weitergabe versuchen, dem immateriellen Gut „Software" Züge eines „normalen" physischen Gutes zu verleihen (Urheberrechtsschutz, Lizenzen, Patente).

Die genannte Problematik ist schwerwiegender im Bereich der Standardsoftware, da Individualsoftware aufgrund ihres individuellen Charakters in vielen Fällen (zumindest als Einheit) nicht direkt für andere Anwender als den ursprünglichen Käufer der Software nutzbar ist, aber auch Teile (einzelne Module oder Komponenten) von Individualsoftware können dieser Art von Marktversagen unterliegen. Ökonomisch gesehen sind somit die „Kopierkosten" von Individualsoftware wegen der anfallenden Informationskosten und Kosten der Anpassung systemimmanent höher als jene von Standardsoftware, wodurch schon von sich aus ein gewisser „monopolistischer Spielraum" und Schutz der Software selbst entsteht.

Generell hat sich in der Praxis die marktkonforme Lösung durch Schutz des geistigen Eigentums mittels Lizenzen durchgesetzt. Dies ist wie oben dargelegt besonders für Standardsoftware von Bedeutung, da diese eher die Charakteristik eines physischen Gutes annimmt, während Individualsoftware oft untrennbar mit einer Dienstleistung (Entwicklung, Wartung, Aktualisierung von Schnittstellen etc.) verbunden ist und damit selbst eher den Charakter einer Dienstleistung annehmen kann. Der rechtliche Schutz von Software zielt somit bewusst darauf ab,

(a) den entwickelnden Unternehmen für innovative Produkte einen Markt zu bieten, in dem das marktbedingte Marktversagen reduziert wird, und

(b) zusätzlich beschränkte monopolartige Spielräume einzuräumen, damit Anreize für Innovation und Forschungsleistung entstehen.

Unabhängig von den verwendeten Lizenzmodellen und auch weitgehend losgelöst vom oben dargestellten Staatseingriff führt die Existenz von Netzwerkeffekten nun dazu, dass der „Wert" von Software mit zunehmender Verbreitung ebenfalls steigt. Ursachen dafür sind in der Bedeutung kompatibler Schnittstellen und der Austauschbarkeit von Daten sowie in einer erwünschten Interoperabilität verschiedener Softwareprodukte zu finden. Der Nutzer eines Programms profitiert also davon, dass auch andere Personen dieselbe Software verwenden – umso höher der Marktanteil eines Produkts, desto größer der zusätzliche Nutzen für den Anwender.

Aus diesem Grund gibt es auf dem Softwaremarkt generell eine starke Tendenz zur Monopolbildung, die ebenfalls systembedingt ist und nicht durch den Staatseingriff oder ein bestimmtes Lizenzmodell ausgelöst wird. So gibt es außer den immer wieder zitierten Beispielen z. B. für bestimmte kommerzielle Betriebssysteme zahlreiche Beispiele für (aufgrund ihrer hohen Verbreitung) monopolartige Software, die sogar auf freiem Quellcode beruht (Open Source Software) und gratis zur Verfügung gestellt wird. Diese Tendenz könnte auch durch die Aushöhlung von Lizenzen nur schwer bekämpft werden, da sie

(a) darauf beruht, dass der tatsächliche gesamtwirtschaftliche Nutzen der verbreiteten Software durch die vermehrte Verbreitung erhöht wird, und

(b) nicht primär von den rechtlichen Rahmenbedingungen hervorgerufen wird, sondern ein Charakteristikum dieses spezifischen Marktes darstellt.

Selbstverständlich birgt diese Monopolisierungstendenz wiederum die Gefahr eines weiteren Marktversagens, da Monopole zu höheren Preisen und ebenfalls zu einer Unterversorgung des Marktes führen können. Als die effektivste Gegenmaßnahme wird in der ökonomischen Theorie einhellig davon ausgegangen, dass potentielle oder tatsächliche Konkurrenz einen Monopolisten dazu bringt, in einer für den Markt effizienten Weise anzubieten – tut er das nicht, droht ihm jederzeit der Verlust seiner beherrschenden Marktposition. Daher ist es erforderlich, einen Wettbewerb zwischen unterschiedlichen Produkten verschiedenster Anbieter und Philosophien (z. B. Lizenzmodelle) zuzulassen, ohne die eine oder die andere Variante in irgendeiner Weise aktiv zu bevorzugen. Der Markt kann seine Aufgabe der effizienten Allokation (Zuteilung der Ressourcen im Produktionsprozess) am besten dann erfüllen, wenn die Entscheidungen der Marktteilnehmer

ausschließlich nach rational-ökonomischen Gesichtspunkten erfolgen, also insbesondere aufgrund technischer Eignung und dem Kosten-Nutzen-Verhältnis der Alternativen. Jegliche Diskriminierung (z. B. nach der Art des Lizenzmodells oder der Organisation der Entwicklungsarbeit) sollte daher im Sinne hoher Markteffizienz unterbleiben, damit der Markt selbst die Entscheidung für oder gegen ein Produkt treffen kann und seine disziplinierende Funktion optimal erfüllen kann.

4.2.3 Die Frage der Unabhängigkeit vom Hersteller

Sehr oft wird für die öffentliche Verwaltung gefordert, dass eine gewisse Unabhängigkeit von einzelnen Herstellern gewahrt werden soll. Oft wird dabei unterstellt, dass Open Source Software jedenfalls diese Anforderung erfüllt.

In der Realität impliziert die Eigenschaft eines Programms als Open Source Software aber nicht, dass die Software auch wirtschaftlich sinnvoll vom Hersteller unabhängig modifiziert oder gewartet werden kann bzw. „unbegrenzt“ haltbar ist:

- Der wirtschaftliche Aufwand der Wartung und Modifikation bestehender Software kann möglicherweise nur vom ursprünglichen Hersteller getragen werden, da dieser einen Informationsvorteil gegenüber den Konkurrenten nützen kann.

- Auch im Open-Source-Bereich kommt es zu Weiterentwicklungen bestimmter Komponenten des Betriebssystems oder anderer Hilfsprogramme. Daher sind „alte“ Versionen von Open Source Software oft auf „neuen“ Plattformen ebenso wenig lauffähig wie veraltete proprietäre Produkte. Überdies kommt es in beiden „Welten“ auch zu einer Veränderung der Hardwareanforderungen und -spezifikationen, sodass auch noch theoretisch lauffähige „alte“ Software alle paar Jahre einer intensiveren (und kostspieligen) Wartung unterzogen werden muss.

Zusammenfassend kann angemerkt werden, dass auch Open Source-Software in der Praxis keine vollkommene Herstellerunabhängigkeit bringen kann. Bei vielen Open-Source-Projekten im öffentlichen Bereich sind große internationale Konzerne involviert, die umfangreichen Programmcode erzeugen, sodass der Umstieg auf einen anderen IT-Anbieter mit großen Übertrittskosten für die öffentliche Hand verbunden wäre.

4.2.4 Arten von Software

Die ökonomischen Mechanismen, die auf dem Softwaremarkt zum Tragen kommen, sind für unterschiedliche Arten von Software nicht identisch. Im Zusammenhang mit der Problematik der Marktunvollkommenheiten des Softwaremarktes lässt sich insbesondere die aktuelle Diskussion über Vor- und Nachteile geschützter Software gegenüber Open Source Software kaum umgehen. Vielfach sind die verwendeten Begriffe allerdings unklar gegeneinander abgegrenzt, weswegen hier zunächst eine kurze Klarstellung erfolgen soll.

Standardsoftware (packaged software): ist in vielen Fällen closed source (Quellcode wird nicht offen gelegt) und proprietär. Standardsoftware verursacht aufgrund der hohen Standardisierung geringe Grenzkosten und kann (technisch) beliebig vervielfältigt werden. Dadurch ergeben sich

große Netzwerkeffekte (Vorteile der Verbreitung einer einheitlichen Technologie, d. h. je mehr Nutzer, desto größer die individuellen Nutzervorteile) und steigende Skalenerträge (sinkende Durchschnittskosten bei größerer Verbreitung, Fixkostendegression). Eine große Verbreitung der Standardsoftware führt zum Phänomen eines impliziten Standards (z. B. kann Software, die für ein verbreitetes PC-Betriebssystem geschrieben wurde, auf fast allen derartigen PCs ohne Probleme installiert werden). Die Duplizierbarkeit der Softwareleistung führt zu hoher Effizienz in Bezug auf die Qualifikationen der User; diese können Kenntnisse von Job zu Job oder auch vom Job ins Privatleben und umgekehrt mitnehmen. Standardsoftware hat eher die Charakteristik eines physischen Produkts, sofern sie dem Urheberrechtsschutz unterliegt (das ist in der Regel der Fall). Es besteht allerdings die Gefahr des Marktversagens (gesamtwirtschaftlich zu geringes Angebot an Software, wenn der Staat nicht durch den Schutz des geistigen Eigentums ein begrenztes Monopol für innovative Unternehmen schafft), daher erfolgt in den meisten westlichen Staaten ein entsprechender Staatseingriff durch Urheberrechte und Patente (analog zu Pharmaindustrie, Musikindustrie, ...).

Open Source Software: ist in vielen Fällen gratis und steht unter verschiedenen Lizenzmodellen zur Verfügung (GPL, BSD, LGPL, MPL, ...), die eine kommerzielle Verwertung in einigen Fällen erlauben, in anderen nicht. Meist handelt es sich um Individualsoftware oder zumindest stark individualisierte Software. Es überwiegt daher oft der Dienstleistungscharakter. Open Source Software mit Individualcharakter ist oft nicht leicht ohne Modifikation in anderen Anwendungen einsetzbar oder muss erst (im Falle von so genannter „Middleware“) an die Kundenbedürfnisse angepasst werden, dafür müssen von Anwendung zu Anwendung konstante Elemente nicht jedesmal neu erfunden werden, sondern stehen der „Community“ als „common knowledge“ zur Verfügung. Durch den häufig beobachtbaren Charakter als Individualsoftware ist Open Source Software insbesondere für nicht-standardisierte Prozesse im Unternehmen im Einsatz, in vielen Fällen allerdings auch im Bereich Internetserver. Je nach genauer Ausgestaltung der Lizenz kommt es zu unterschiedlichen Arten von Marktversagen:

(a) eine Unterversorgung mit qualitativ hochwertiger Software für bestimmte Anwendungen (z. B. benutzerfreundliche Betriebssysteme für End-User) entsteht im Falle sehr strikter Open Source Software-Lizenzen (z. B. GPL – ist der überwiegende Teil von Open Source Software), die die kommerzielle Nutzung der Komponenten vollkommen verbieten;

(b) eine Überversorgung mit auf Open Source basierender Individualsoftware ist ebenfalls möglich, da der Entwickler auf einen Gratis-Pool zurückgreifen kann und die Kosten der Entwicklung von jemandem anderen getragen werden mussten, der für seine Leistung jedoch nicht ausreichend ökonomisch belohnt wurde. Oft setzen aber auch bei Open Source Software Marktmechanismen ein, indem solche Unternehmen komplementäre Güter und Services verkaufen (Hardware, Software, Dienstleistungen) und damit Gewinne erzielen. Diese Dienstleister sind sehr oft hochgradig vertikal integriert, bieten also Produkte und Leistungen auf unterschiedlichen Produktionsstufen an (z. B. Hardware, Betriebssystem, Support, Training, Software-Entwicklung, ...).

4.2.5 Ökonomische Funktionsweise von Open Source Software

Im Zusammenhang mit freiem Quellcode ist eine der zentralen Fragen, wie der Open-Source-Sektor in ökonomischer Hinsicht funktioniert. Die grundlegende Philosophie der Entwicklung von Open Source liegt in der dezentralen Entwicklung von Software durch räumlich in der Regel stark verstreute Entwicklungsgemeinschaften. Auf der anderen Seite ist auch die Entwicklung von Open Source Software in kommerziellen Unternehmen zu beobachten. Generell muss Open Source Software in einem ersten Schritt danach unterschieden werden, ob überhaupt eine ökonomische Aktivität erfolgt oder nicht:

Wird solche Software von einem Programmierer in seiner Freizeit (unbezahlt) entwickelt und in Umlauf gebracht[1], so entstehen dadurch keine ökonomisch messbaren Transaktionen, d. h. es wurde keine direkte oder indirekte Wertschöpfung (in der Softwarebranche oder in vorgelagerten Branchen) erzielt, ebenso wird keine sekundäre Wertschöpfung erfolgen (durch Kaufkrafteffekte), da der Programmierer ja kein Entgelt erzielt. Lediglich Anwendungseffekte können bei entsprechender Nutzbarkeit und Verbreitung des Codes auftreten, die je nach der Wiederverwendbarkeit unterschiedlich hoch sein können.

Auf der anderen Seite gibt es Unternehmen, die Open Source Software kommerziell entwickeln und anbieten bzw. auch Teile ihrer Entwicklungen unentgeltlich frei zugänglich machen. In diesem Fall stellt sich die Frage, wieso diese Unternehmen freiwillig darauf verzichten, den Schutz ihres ideellen Eigentums in Anspruch zu nehmen. In der Literatur werden dazu unterschiedliche Gründe genannt:

- Die kommerziell entwickelte freie Software ist ein Komplementärgut zu weiteren ökonomischen Aktivitäten (z. B. Hardware, Anpassung der vorhandenen Software an die Kundenbedürfnisse, Dienstleistungen im Bereich Ausbildung, Training, Wartung, Support, ...). Die Entwicklungskosten solcher Software werden durch die „Kuppelprodukte“ quersubventioniert. Auf diese Weise wird Open Source Software letztlich doch verkauft, allerdings über den Umweg anderer Produkte.

- Open Source Software wird als Marketingargument im Verkauf von komplementären Leistungen eingesetzt, da diese derzeit in der öffentlichen Wahrnehmung äußerst positiv besetzt ist und bestimmte Eigenschaften des Produkts zu Recht oder zu Unrecht suggeriert.

- Die Integration vorhandener Open-Source-Komponenten erfordert eine hohe fachliche Qualifikation. Da die Einarbeitung in IT-Systeme von Fremdherstellern (insbesondere bei Individualsoftware und im Falle einer Vielzahl konkurrierender Open-Source-Projekte mit ähnlicher Funktionalität) mit erheblichem Aufwand verbunden sein kann, entsteht auf diesem Umweg ein monopolistischer Schutz der kommerziellen Anbieter von OSS-Lösungen, da alternative Dienstleister erheblichen Aufwand bei der Übernahme von Kunden hätten und daher die Umstiegskosten für den Kunden hoch wären.

[1] Die Abhandlung der Motive für Programmierer, Software unentgeltlich herzustellen, würde den Rahmen dieser Arbeit sprengen. Auch ist in der OSS-Community umstritten, ob der OSS-Programmierer von seiner Tätigkeit einen ökonomisch messbaren Vorteil erzielt oder nicht.

Im Falle dieser kommerziellen Verwertung von Open Source Software können die anbietenden Unternehmen entweder zusätzliche (monopolartige) Gewinne realisieren oder versuchen, die Marktpreise zu unterbieten. In beiden Szenarien sind die direkten und indirekten Effekte in der ersten Produktionsstufe jedoch geringer als bei geschützter Software[2].

4.3 KOSTENDIMENSION (TCO) UNTERSCHIEDLICHER ALTERNATIVEN

Zur Kostendimension von OSS Software ist eine ganze Reihe von unterschiedlichen Arbeiten in der Literatur erschienen.

Laut vorhandenen Studien und eigenen Erhebungen beträgt der Anteil der Softwarekosten an den gesamten IT-Kosten durchschnittlicher österreichischer Unternehmen (total cost of ownership – TCO) rund 4 bis 5%, die weitaus größeren Teile entfallen auf Personalkosten (60 bis 65%) sowie Ausfallskosten (20 bis 25%). Hardwarekosten und Kosten der Ausbildung qualifizierter Mitarbeiter schlagen sich jeweils mit rund 5% zu Buche. Es gibt keine Hinweise darauf, dass österreichische Unternehmen in dieser Hinsicht signifikant vom europäischen Schnitt abweichen.

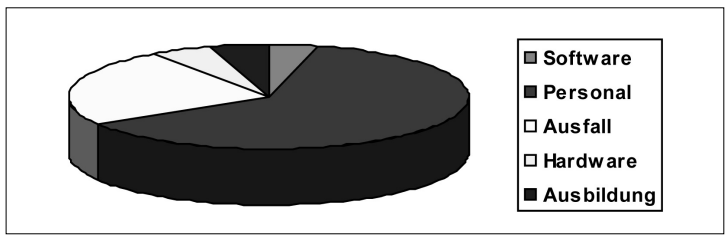


Abbildung 1: Komponenten der TCO bei IT-Anwendungen (Österreich). Quelle: eigene Erhebungen (2003).

Die Ausbildungskosten für IT-Personal machen im durchschnittlichen österreichischen Unternehmen pro ArbeitnehmerIn rund 1 Monatsgehalt pro Jahr aus. Die raschen Änderungen der Anforderungen an das Qualifikationsprofil der Arbeitskräfte und die ohnehin hohen betrieblichen Ausgaben für Qualifizierungsmaßnahmen sprechen für eine Stärkung der IT-Grundausbildung im Rahmen arbeitsmarktpolitischer Maßnahmen. Da Open Source Software derzeit allerdings im Office-Bereich und auf PCs/Workstations noch kaum verbreitet ist, ergibt sich keine ökonomische Notwendigkeit für einen forcierten aktiven Ausbildungsimpuls seitens der Arbeitsmarktpolitik. Im Falle ausreichender Nachfrage von der Unternehmensseite sind jedoch (ergänzende) Schulungen mit entsprechenden Open-Source-Tools in Abstimmung mit den Marktbedürfnissen zu begrüßen.

Bei der Einführung von Open Source werden nun die Softwarekosten zu einem bestimmten Teil eingespart. Es kommt jedoch zu verstärkten Kostenbelastungen im Bereich von Training und Support, sodass insgesamt keine nennenswerten Einsparungspotenziale genützt werden können.

[2] Es wäre vollkommen falsch anzunehmen, dass (auch unbezahlte) OSS keinen gesamtwirtschaftlichen Effekt hat, so wie auch z. B. unbezahlte Hausarbeit wertvolle Folgeeffekte auslöst, die sich auch ökonomisch niederschlagen. In beiden Fällen entfällt jedoch im ersten Schritt der Leistungserstellung der Produktionseffekt. Bezahlte Hausarbeit hätte größere ökonomische Effekte (im Sinne der Systematik der Volkswirtschaftlichen Gesamtrechnung, VGR) als unbezahlte, Analoges gilt für unbezahlte und bezahlte Softwareentwicklung.

Besonders ausführlich wurde anhand des Beispiels der Stadt München im Rahmen der so genannten „Unilog-Studie“ analysiert, welche technischen und wirtschaftlichen Effekte vom Umstieg auf Open-Source-Produkte ausgehen könnten. Es existiert in diesem Bereich jedoch noch eine Fülle weiterer Untersuchungen mit widersprüchlichen Resultaten. Als wichtigste Schlussfolgerungen für die TCO können dabei folgende Hauptargumente genannt werden:

- Die Einführung von Open Source Software ist mit erheblichen Entwicklungskosten verbunden, die erst über einen sehr langen Zeitraum von den geringeren Lizenzkosten als bei proprietären Produkten kompensiert werden können (in der Literatur werden dabei Zeiträume in der Größenordnung von 10 Jahren genannt).

- Die Einschulungs-, Trainings- und Supportkosten machen in einigen Fällen bei Open Source Software mehr als 50% der gesamten IT-Kosten aus.

- Die Integration unterschiedlicher Plattformen in ein konsistentes System ist mit zusätzlichem Aufwand verbunden.

- Wenn das erforderliche Open-Source-Know-how intern vorhanden ist, können bestimmte isolierte Anwendungen (z. B. im Webserverbereich) relativ kostengünstig auf dieser Basis implementiert werden.

- Auch für Open-Source-Arbeitsplätze, die mit Open Source Software ausgestattet sind, fallen bei kommerziellen Distributionen Kosten pro Arbeitsplatz an, die sich in etwa in der Größenordnung proprietärer Produkte bewegen, sofern die unterschiedlichen Paletten von Funktionen berücksichtigt werden.

- Geringer Bedarf an integrierten Lösungen auf Ebene der gesamten Organisation fördert tendenziell Kostenvorteile bei Open-Source-Produkten – integrierte Lösungen wiederum verursachen weniger technischen und wirtschaftlichen Aufwand im Bereich der proprietären Lösungen.

- Je höher der Grad der Standardisierung der Anforderungen an die verwendete Software, desto eher wird Standardsoftware geringeren Aufwand verursachen als Open Source Software.

- Müssen spezifische Anwendungen erst entwickelt werden, halten sich die Kostenvorteile der beiden Alternativen je nach Betrachtungsweise oftmals die Waage.

Eine Entscheidung auf Basis der TCO allein ist kaum zufrieden stellend möglich. Es müssten unterschiedliche Funktionalitäten der einzelnen Lösungen berücksichtigt werden.

4.4 SOFTWARE ALLGEMEIN – EINE „GUTE“ INVESTITION DES ÖFFENTLICHEN SEKTORS?

Für die Beantwortung der Frage, ob der Staat bestimmte Arten von Software aktiv fördern sollte, muss zunächst überprüft werden, ob Investitionen in den Bereich Software durch die öffentliche

Hand überhaupt ökonomisch wünschenswerte Effekte nach sich ziehen. Obwohl sich die Definition von „wünschenswert“ im Kontext wirtschaftspolitischer Entscheidungen nur in einem normativen Zusammenhang (nicht wertfrei) behandeln lässt, wird in der ökonomischen und wirtschaftspolitischen Literatur in der Regel auf grundlegende gesamtwirtschaftliche Größen, wie die Wertschöpfung (Beitrag zur gesamten Wirtschaftsleistung eines Landes) oder die Auswirkungen auf die Beschäftigung abgestellt.

Unabhängig davon, dass verstärkter IT-Einsatz im öffentlichen Sektor zu betriebswirtschaftlichen Einsparungen, Effizienzsteigerungen, qualitativ höherwertigen Dienstleistungen den Bürgern gegenüber sowie einer Vielzahl von weiteren positiven Effekten führen kann, muss in einem ersten Schritt der Stellenwert des IT-Sektors in der österreichischen Wirtschaft geklärt werden.

Die Softwarebranche insgesamt (Softwarehäuser und Datenverarbeitungsdienste sowie Datenbanken) beschäftigt rund 32.000 Personen (entsprechend ca. 20.500 Vollzeitäquivalenten). Etwa 4.800 Softwarehäuser und 2.800 Unternehmen im Bereich Datenverarbeitungsdienste sowie weitere Unternehmen (insgesamt ca. 8.000 Unternehmen) erzielen dabei einen Nettoumsatz von jährlich rund 5,5 Mrd., wobei rund 1,5 Mrd. auf den Bereich Standardsoftware entfallen.

Beschäftigte (Arbeitsplätze)	32.000 Personen
Beschäftigte (Vollzeitäquivalente)	20.500 Personen
Unternehmen gesamt	8.000 Unternehmen
➛ davon Softwarehäuser	4.800 Unternehmen
➛ davon Unternehmen im Bereich Datenverarbeitungsdienste	2.800 Unternehmen
Nettoumsatz	5,5 Mrd. Euro
➛ davon Standardsoftware	1,5 Mrd. Euro
➛ davon IT-Dienstleistungen	3,4 Mrd. Euro
➛ davon sonstige Software	0,6 Mrd. Euro

Tabelle 1: Rahmendaten des Softwaresektors

Der Softwaresektor (insgesamt) in Österreich wendet pro Beschäftigtem rund 44.000 an Personalausgaben pro Jahr auf und belegt somit Platz 2 im Vergleich zu allen anderen Dienstleistungsbranchen (nach dem Bereich „Forschung und Entwicklung“ mit 47.000). Vom Umsatz her gehen rund 90% an andere Unternehmen, 9% an den öffentlichen Sektor und 1% an private Haushalte. Zwischen 1976 und 2000 ist der Beitrag zur gesamtösterreichischen Wertschöpfung um das rund 15fache gestiegen. Software und Datenbanken haben zusammen ein Gewicht von über 4% im Preisindex der Ausrüstungsinvestitionen von Unternehmen mit in den letzten Jahren steigender Tendenz.

Das hohe Einkommen der Beschäftigten im Softwaresektor führt dazu, dass wirtschaftliche Aktivitäten im Softwaresektor (siehe unten) starke Kaufkrafteffekte nach sich ziehen und somit die gesamtwirtschaftlichen Auswirkungen öffentlicher Investitionen in Software sich keinesfalls auf den Softwaresektor und die dort beschäftigten Personen beschränken, sondern darüber hinaus Multiplikatoreffekte zu beobachten sind, die auch alle anderen Branchen betreffen.

Aus ökonomischer Sicht führt der verstärkte Einsatz von Software im Zusammenhang mit öffentlichen Aufträgen und Investitionen zu einer Erhöhung der Wertschöpfung und zu einer Verbesserung der Beschäftigungssituation.

4.5 GESAMTWIRTSCHAFTLICHE WIRKUNGSMECHANISMEN

Bei der Betrachtung der gesamtwirtschaftlichen Effekte öffentlicher Aktivitäten (im konkreten Fall: Auftragsvergabe an die Softwarebranche) wäre es unvollständig, die Auswirkungen dieser Maßnahmen lediglich über die so genannte „Bruttoinvestitionssumme" (Auftragsvolumen) zu messen. Aus ökonomischen Modellen können zu diesem Zweck „Multiplikatoren" berechnet werden, die den unmittelbar auf die Softwarebranche einwirkenden Effekt (neue Arbeitsplätze, mehr Wertschöpfung) in ein Verhältnis zum gesamtwirtschaftlich zu beobachtenden Effekt setzen.

Diese Multiplikatoren zeigen an, welche direkten Effekte (Wertschöpfung und Beschäftigung in jenen Unternehmen, die direkt in der Softwarebranche tätig sind) aus einer Erhöhung der Nachfrage nach den entsprechenden Gütern entstehen. Die indirekten Effekte beschreiben, welche Wertschöpfungs- und Beschäftigungseffekte in jenen Unternehmen entstehen, die Vorleistungen (Vorprodukte, Betriebsstoffe usw.) zuliefern. Die Summe aus direkten und indirekten Effekten ergibt die so genannten „primären Effekte" („First-round Effects"). Die „sekundären Effekte" entstehen dadurch, dass private Haushalte (u. a. die direkt und indirekt Beschäftigten) aus diesen Aktivitäten Einkommen beziehen (Teil der direkten und indirekten Wertschöpfung), welches (nach Abzug der einbehaltenen Ersparnisse) für Konsumgüter ausgegeben wird („Second-round Effects").

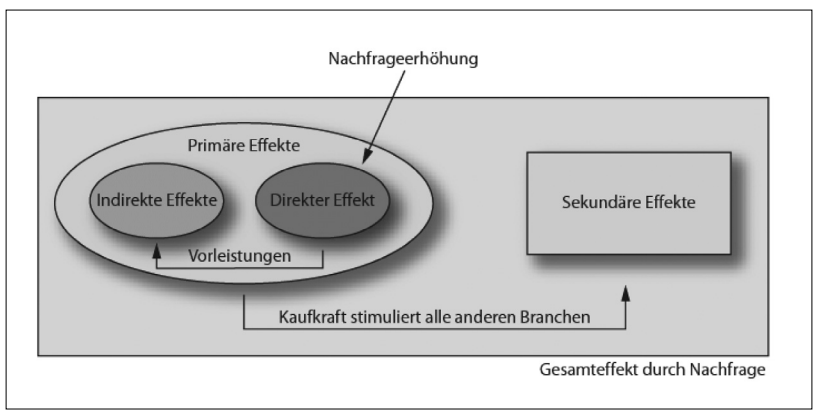


Abbildung 2: Struktur der gesamtwirtschaftlichen Effekte von Softwareinvestitionen

4.6 MULTIPLIKATOREN DER UNTERSCHIEDLICHEN SOFTWARESEGMENTE

Eine analytisch einwandfreie Trennung im statistischen Material für Österreich zwischen Open Source Software und proprietärer Software ist nicht ohne Schwierigkeiten möglich, die Trennung in Standardsoftware und Nicht-Standardsoftware (sonstige Software, Individualsoftware) hingegen durchführbar. Daher werden die Multiplikatoren für diese Kategorien dargestellt, wobei

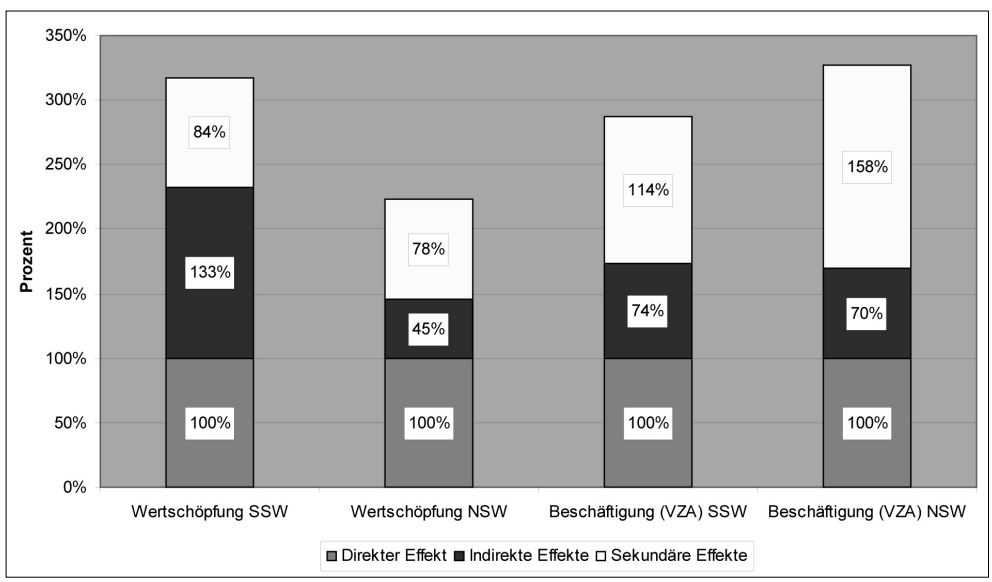
Standardsoftware in der Regel mit proprietärer Software korreliert ist, während Open Source Software eher in die zweite Kategorie fällt (siehe Ausführungen weiter oben).

Auf dem derzeitigen Aktivitätsniveau der Softwareindustrie in Österreich betragen die durchschnittlichen (Produktions-)Multiplikatoren für Standardsoftware derzeit rund 1 : 4,5 für die Arbeitsplätze (Effekt eines neuen Arbeitsplatzes in der Softwarebranche auf den gesamten Arbeitsmarkt) und rund 1 : 3,2 für die Wertschöpfung (Auswirkung einer Erhöhung der Wertschöpfung innerhalb der Softwarebranche auf die Wertschöpfung der Gesamtwirtschaft). Für sonstige Software betragen die Multiplikatoren 1 : 5,1 für die Beschäftigung und 1 : 2,2 für die Wertschöpfung (Tabelle 2).

	Beschäftigungsmultiplikator		Wertschöpfungsmultiplikator
	Vollzeitäquivalente	Arbeitsplätze	
Standardsoftware ohne Services	2,88	4,52	3,17
Sonstige Software	3,27	5,14	2,23

Tabelle 2: Produktionsmultiplikatoren von Software

Abbildung 3 stellt die Aufteilung der Gesamteffekte auf die einzelnen Komponenten grafisch dar. Diese Effekte werden jedoch lediglich durch die Produktionstätigkeit der Softwarebranche hervorgerufen. Bezieht man noch Effekte der Anwendung von Software in den übrigen Sektoren der Wirtschaft mit ein, so ergeben sich darüber hinaus noch zusätzliche Multiplikatoren, die wiederum je nach Standardisierung und Verbreitung eines Produkts bei Standardsoftware bis zu 1 : 35 für die Beschäftigung und 1 : 8 für die Wertschöpfung betragen und sich bei Individualsoftware in Größenordnungen von 1 : 3 für die Beschäftigung bzw. 1 : 2 für die Wertschöpfung bewegen.



*) SSW: Standardsoftware, NSW: Nicht-Standardsoftware, VZA: Vollzeitäquivalente

Abbildung 3: Wirkungsmultiplikatoren (Wertschöpfung und Beschäftigung)

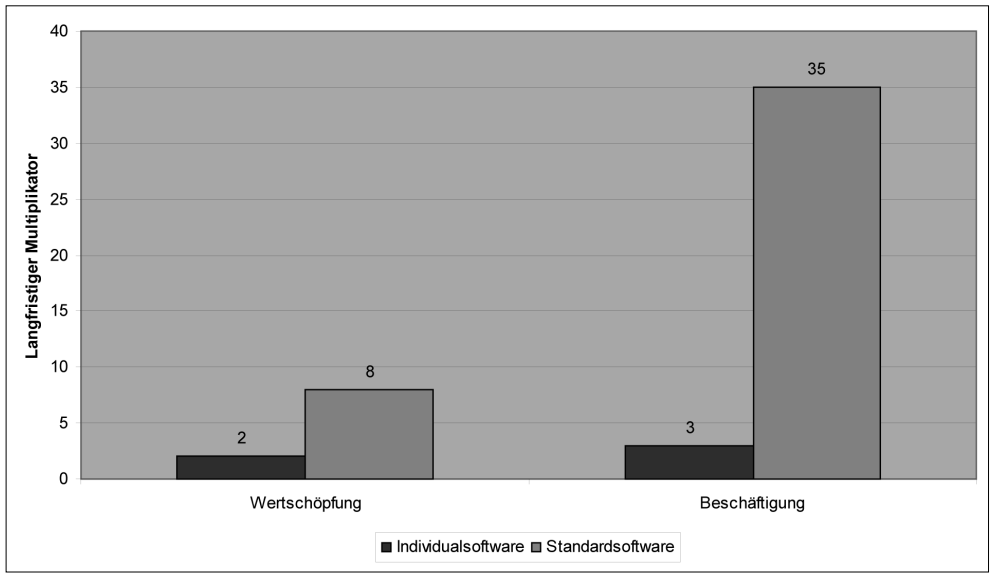


Abbildung 4: Anwendungsmultiplikatoren (Wertschöpfung und Beschäftigung)

Unter Einbeziehung aller Effekte (Produktionseffekte, Anwendungseffekte) ergibt sich für höchstgradig standardisierte Software ein impliziter durchschnittlicher Gesamtmultiplikator von ca. 1 : 11 für die Wertschöpfung. Aufgrund der Inhomogenität des Bereiches Individualsoftware und IT-Services kann für diesen anderen Bereich kein verlässlicher Gesamtmultiplikator errechnet werden, dieser wäre jedoch (wie oben dargestellt) geringer.

4.7 WIRTSCHAFTSPOLITISCHE SCHLUSSFOLGERUNGEN

Die dargestellten Überlegungen zeigen auf, dass Software überhaupt als „Technologie“ oder „Information“ verschiedenen Arten von Marktversagen unterliegt, insbesondere durch die leichte Kopierbarkeit („Nicht-Ausschließbarkeit“) und die Präsenz externer Effekte (Netzwerkeffekte, Nutzen für andere Marktteilnehmer, der nicht vom Markt abgegolten wird). Während im Bereich der proprietären (Standard-)Software ersteres Problem vom Staat durch Urheberrechte gemildert wird (also „Ausschließbarkeit“ geschaffen wird, die bei anderen Gütern selbstverständlich ist oder in anderen vergleichbaren Branchen in ähnlicher Weise herbeigeführt wird), wird gerade diese freie Verfügbarkeit definitionsgemäß als Charakteristikum von Open Source Software gesehen, daher scheidet ein Schutz des geistigen Eigentums für solche Software definitionsgemäß aus.

Ob nun Open Source Software staatlich zu fördern ist oder nicht, muss anhand von zwei möglichen Szenarien geprüft werden:

a) Wenn Open Source Software den proprietären Lösungen technisch und wirtschaftlich überlegen ist, so ist kein fördernder Staatseingriff erforderlich, da sich die bessere Technologie am Markt (unter sonst gleichen Wettbewerbsbedingungen) ohnehin durchsetzen wird.

b) Ist Open Source Software aber wegen eines Marktversagens (durch Nicht-Ausschließbarkeit und externe Effekte) nicht ausreichend am Markt vorhanden, dann legt die ökonomische Theorie nahe, dies durch einen Staatseingriff auszugleichen – dieser Staatseingriff wäre in der Regel durch einen Schutz des geistigen Eigentums zu erzielen, würde also aus OSS proprietäre Software machen! Eine Form der Subventionierung oder positiven Diskriminierung zur Abgeltung der Netzwerkeffekte wäre ebenfalls schwer zu rechtfertigen, da diese auch bei proprietärer Software auftreten und nicht vom zugrunde liegenden Entwicklungs- oder Geschäftsmodell abhängen.

Ein weitergehender aktiver Staatseingriff, der die eine oder andere „Software-Welt" bevorzugt, erscheint daher derzeit nicht geboten (und wäre voraussichtlich mit vergleichsweise höheren sozialen Kosten verbunden). Insgesamt wäre es für den öffentlichen Sektor ratsam, die IT-Investitionen jeweils auf den Einzelfall bezogen anhand von objektiven Kosten-Nutzen-Überlegungen in Bezug auf die Software selbst zu bewerten, wie dies bei jedem wirtschaftlich orientierten Unternehmen auch der Fall ist (siehe dazu auch die diesbezüglichen Empfehlungen und Richtlinien des IKT-Boards der Bundesregierung). Eine Wettbewerbsverzerrung (z. B. durch Bevorzugung oder Förderung) zugunsten des einen oder anderen Geschäftsmodells der IT-Welt würde die Marktmechanismen behindern und hätte insbesondere im Bereich Open Source Software keine großen (positiven) ökonomischen Effekte, da

(a) im Falle nicht-gewinnorientierter Open Source Software ökonomische Anreize mangels ökonomischer Interessen de facto unwirksam sind bzw.

(b) im Falle kommerzieller (z. B. vertikal integrierter) Anbieter die Bevorzugung von Open Source Software (z. B. als Beiwerk zu Hardware oder Dienstleistungen mit Gewinnabsicht) wirtschaftspolitisch nicht begründbar wäre und zu einer ökonomisch ineffizienten Überversorgung mit den gekoppelten Produkten führen würde.

Große Bedeutung kommt jedoch im Gegensatz zur weitgehenden Irrelevanz des Entwicklungsmodells dem Bereich der klar definierten (bzw. „offenen") Schnittstellen und Standards zu. Dadurch wird sichergestellt, dass alle Anbieter von Software die gleichen Chancen haben und der Markt seine Aufgaben effizient erfüllen kann. Überdies stellt deren Einhaltung für den Käufer bzw. Auftraggeber sicher, dass der Austausch einzelner Module des Informationssystems mit verhältnismäßig geringerem Aufwand möglich ist, es also zu keiner Verstärkung langfristiger Abhängigkeiten kommt (die durch das übliche Volumen der Investitionen in den betrieblichen bzw. öffentlichen IT-Bereich und die voraussichtliche Nutzungsdauer über mehrere Jahre in Form von „sunk cost" ohnehin unvermeidbar sind). Eine marktkonforme Lösung mit möglichst geringen aktiven („diskretionären") Eingriffen des öffentlichen Sektors in den IT-Markt wird von der überwältigenden Mehrheit der Ökonomen als überlegen eingestuft.

Die Entscheidung für Open-Source-Applikationen oder proprietäre Lösungen sollte demnach auch im öffentlichen Sektor oder Non-Profit-Bereich nicht in erster Linie vom Entwicklungsmodell oder Geschäftsmodell abhängig gemacht werden, sondern primär nach technischen Aspekten und dem Preis-Leistungs-Verhältnis erfolgen.

Kapitel 5

RECHTLICHE ASPEKTE EINES OPEN SOURCE SOFTWARE-EINSATZES IN DER ÖFFENTLICHEN VERWALTUNG

5.1 EINLEITUNG

Im Bereich der öffentlichen Verwaltung steht eine Reihe so genannter E-Government-Projekte an. Vorrangige Ziele dabei sind die Vereinheitlichung, Interoperabilität sowie die Zuverlässigkeit unterschiedlicher Behördensysteme. Entscheidendes Kriterium dabei ist die vorangehende Wahl der Softwarelizenzierung.

Am 09.11.2001 fasste der deutsche Bundestag den Beschluss „zur Förderung von Open Source in der deutschen Verwaltung". In Ausführung des Beschlusses entschied sich der Ältestenrat des deutschen Bundestages am 14.03.2002 dafür, innerhalb der Bundestagsverwaltung im Serverbereich Linux zu installieren. Clientseitig beließ man es bei Microsoft-Produkten. Trotz des „unvollständigen" Wechsels hin zu Open Source kam dieser Entscheidung eine absolute Signalwirkung zu.

In Österreich fasste das IKT[1]-Board am 25.06.2002 einen Beschluss zum Thema „Alternativen zu monopolartiger Software"[2]. Es handelt sich dabei um einen Vorschlag, in Hinkunft bei der Anschaffung von Servern sicherzustellen, dass diese auch unter Linux betrieben werden können (Ausschreibungsbedingung). Die Anschaffung solcher Server soll nicht mit Mehrkosten verbunden sein, zumal sie die Basis der Umsetzung des Beschlusses sichern soll. Am 17.09.2002 folgte der Beschluss, dass in den höheren Sicherheitsklassen Linux als Ausweichsystem verbindlich vorzusehen ist.

Auch auf europäischer Ebene ist man nicht untätig geblieben: im Rahmen des Aktionsplanes **eEurope 2005** wurde bereits auf der Tagung des Europäischen Rates am 19. und 20.07.2000 in Feira der Einsatz von Open Souce Software für die Bereiche „Sicheres Internet und intelligente Chipkarten" sowie „Regieren im Netz – elektronischer Zugang zu öffentlichen Diensten" als strategisches Mittel gesehen und die Mitgliedstaaten wurden aufgefordert, Open Source Software zu fördern.

5.2 POLITISCHE ÜBERLEGUNGEN ZU OPEN SOURCE

Microsoft beherrscht derzeit nach wie vor den Markt sowohl für Betriebssysteme als auch für Textverarbeitungs- und Tabellenkalkulationsprogramme. Faktisch gab es jahrelang keinen Wettbewerb unter den Herstellern, was den Einsatz ihrer Produkte im öffentlichen Sektor betraf. US-amerikanische und europäische Wettbewerbshüter sahen sich in einer solchen Situation auf den Plan gerufen, was dazu führte, dass sowohl in den Vereinigten Staaten als auch in Europa Kartellrechtsklagen gegen den Marktführer Microsoft anhängig waren bzw. noch sind. Im Wesentlichen geht es dabei um die marktbeherrschende Stellung von Microsoft im Bereich der Privatwirtschaft sowie in der öffentlichen Verwaltung. Neben kartellrechtlichen Fragen geht es aber auch um allgemeine strukturpolitische Überlegungen hinsichtlich des IT-Standortes Europa: so stellt der bereits erwähnte Bundestagsbeschluss fest, dass Open Source eine besondere Chance für die europäische Softwarebranche darstelle, da hier zum ersten Mal die USA nicht führend seien.[3]

[1] Informations- und Kommunikationstechnologie
[2] siehe http://www.opensource.at/bmwa0/Public/os_studie/pressemappe_16.06.0/open_source_und_egov
[3] Siehe Heckmann in Spindler „Rechtsfragen bei Open Source", S. 285

Aufgrund dieser politischen und wirtschaftlichen Brisanz des Themas, der Notwendigkeit der öffentlichen Verwaltung, trotz angespannter Haushaltslage die Modernisierung der Verwaltung voranzutreiben, kann man durchaus von einem regelrechten „Open-Source-Boom“ sprechen. Es ist daher seitens der Verwaltung die Möglichkeit in Erwägung zu ziehen, sich über eine IT-Migration Gedanken zu machen. Beispiele für den Einsatz von Open Source Software in der Verwaltung sind die deutschen Städte München[4] und Schwäbisch Hall[5], die Polizei von Niedersachsen, das deutsche Bundesamt für Finanzen, die mexikanischen Schulen sowie die Stadtverwaltung von Mexiko-City.

Linux als inzwischen prominenteste Software mit offenen Quellen gilt als zuverlässiger als die meiste proprietäre Software und dies bei deutlich geringeren Kosten, denn die Entwickler von Open Source Software haben ein Selbstverständnis, welches eher der wissenschaftlichen Arbeitsweise von Universitäten entspricht: im Hinblick auf ein Resultat, das der Allgemeinheit dienen soll, tauschen sich die Softwareentwickler untereinander aus und überprüfen gegenseitig ihre Ergebnisse. Die Open Source Software wird von ihnen als Gemeingut betrachtet, als eine Infrastruktur der Informationsgesellschaft, auf die erst der Wettbewerb aufsetzen soll.

5.3 OPEN SOURCE UND ÖFFENTLICHE VERWALTUNG

Die Verwendung von Open-Source-Produkten in der öffentlichen Verwaltung unterscheidet sich aufgrund ihrer Aufgaben und Funktionen von einem Einsatz im Bereich privater Unternehmen. Die folgenden Problemkreise sind vor einer Heranziehung von Open Source Software besonders zu berücksichtigen.

5.3.1 Das Sicherheitsproblem

Der öffentlichen Verwaltung kommt im Vergleich mit der Privatwirtschaft eine höhere Verantwortung im Hinblick auf eine sichere IT-Technologie zu. Neben einer subjektiven Sicherheit, einer Vertrauenswürdigkeit des Systems, die von den Beurteilungen und Erfahrungen des Nutzers abhängig ist, kommt es beim Einsatz von IT-Systemen vor allem auf eine tatsächliche, objektiv validierbare Sicherheit[6] an. In diesem Zusammenhang spielt die mit Open Source Software verbundene Offenlegung des Codes und die in aller Regel damit einhergehende Nutzung einer Open-Source-Lizenz unter sicherheitstechnischen Aspekten eine entscheidende Rolle. Es wird zwar damit argumentiert, dass die Verwendung von Open-Source-Produkten dazu beitrage, Monostrukturen und damit Abhängigkeiten in der IT-Landschaft zu vermeiden, womit sie auch einen Beitrag zum Systemdatenschutz leiste. Der offene Quellcode gewährleiste eine höhere Anpassungsflexibilität im Falle von Sicherheitslücken.

Auf der anderen Seite erleichtert gerade der offene Quellcode potentiellen Störern einen leichteren Zugriff auf sensible Daten.[7] Die Vorteile eines transparenten Quellcodes lassen sich durch

[4] Die Migration beginnt im Herbst 2004
[5] Die Migration soll im Oktober 2004 abgeschlossen sein, vgl http://www.heise.de/newsticker/meldung/46974
[6] Art. 20 Abs 3 B-VG
[7] Nach einer Studie des Tocqueville-Institutes werden Terroristen geradezu eingeladen, die IT- Infrastruktur eines Landes mit geeigneten Maßnahmen lahmzulegen

die Verwaltung demnach nur in dem Maße nutzen, wie er auch tatsächlich von Fachkundigen beherrscht und ggf. gegenüber Angriffen und Störungen abgeschirmt wird. Es ist davon auszugehen, dass es im Bereich der öffentlichen Verwaltung an einer entsprechenden Kompetenz fehlen wird, so dass man das Know-how extern zukaufen muss. Das „ehrenamtliche“ Engagement der Open-Source-Community kann helfen, Sicherheitslücken zu schließen, die IT-Sicherheit der Behörde kann dies nicht gewährleisten.

5.3.2 Das Abhängigkeitsproblem

Eines der Ziele bei der Verwendung von Open-Source-Produkten ist die Vermeidung einer IT-Monostruktur und damit die Vermeidung von Abhängigkeiten hinsichtlich eines Anbieters[8]. Mangels eigener Entwicklungskompetenz wird die öffentliche Verwaltung auch bei Open Source auf externen Sachverstand zugreifen müssen und so alte Abhängigkeiten durch neue ersetzen. Natürlich hat dies auch seine positive Seite: es ist zu erwarten, dass der Softwaremarkt in Bewegung gerät. Dies vor allem dadurch, da sich mittlerweile zahlreiche Unternehmen auf Open-Source-Produkte spezialisiert haben.

5.3.3 Das Urheberrechtsproblem

Zwei Vorfälle zeigen aber, dass es bei Open Source Software rechtliche Unklarheiten gibt. SCO[9] hat in den USA unter anderem IBM auf Schadenersatz geklagt. IBM wird vorgeworfen, bei Teilen des Linux-Kernels Urheberrechte von SCO verletzt zu haben. In Deutschland hat der VSI[10] mit einem Gutachten zu Open Source Software aufhorchen lassen. Es kommt zu dem Schluss, dass der Einsatz von Open Source Software aufgrund der Lizenztypen wie der GPL rechtlich unsicher sei. Zwar vertritt der MIT-Professor Randall Davis die Ansicht, dass die von SCO beanstandeten Teile im Linux-Kernel nicht von Unix stammen[11], und wurde die rechtliche Wirksamkeit der GPL vom Landgericht München[12] bestätigt, doch sind die Probleme dadurch noch lange nicht im Sinne einer umfassenden und für die öffentliche Verwaltung umso wichtigere Rechtssicherheit gelöst.

5.3.4 Das Patentproblem

Am 18. 05.2004 hat der Ministerrat der EU den Richtlinienvorschlag zur Patentierbarkeit computerimplementierter Erfindungen nach 21 vorgenommenen Änderungen angenommen. Nun muss nur noch das Europäische Parlament über diesen Entwurf abstimmen.

Open Source Software kann Patent- und Gebrauchsmusterrechte in folgenden Bereichen berühren:

- Open Source Software kann bestehende Patentrechte auf proprietäre Software verletzen
- Softwarehersteller oder Programmierer melden Open Source Software als Patente an

[8] Siehe Heckmann in Spindler „Rechtsfragen bei Open Source“, S. 301
[9] Santa Cruz Operation
[10] Verband der Software Industrie
[11] http://derstandard.at/?url=/?id=1748806
[12] http://www.heise.de/newsticker/meldung/49377

Aufgrund dieser Überlegungen wurde das bekannte LiMux-Projekt zunächst auf Eis gelegt, da dieses Projekt etwa 50 Patente verletzen würde.[13] Obzwar ein anschließend in Auftrag gegebenes Rechtsgutachten „nur geringe rechtliche Risiken" bescheinigte, kann keine gänzliche Entwarnung gegeben werden.

5.3.5 Das Wettbewerbsproblem

Als besonders problematisch wird die Weiterentwicklung von Open Source Software durch die Verwaltung, die gemäß der GPL wieder der Öffentlichkeit zur Verfügung gestellt werden muss, angesehen. Somit käme es zu einer „Verstaatlichung" der Softwareindustrie, die vom Steuerzahler getragen würde.[14]

5.3.6 Das Haftungsproblem

Das Haftungsproblem ist eng mit den bisher besprochenen Problemen verbunden. Da bei Open Source Software – jedenfalls beim isolierten Erwerb[15] – nur rudimentäre Gewährleistungspflichten bestehen, wird die Verwaltung verpflichtet sein, entsprechende externe Beratungsleistungen und Services einzukaufen, wenn sie sich nicht eines Organisationsmangels schuldig machen möchte. Hier kann und muss die öffentliche Verwaltung ihre ganze „Macht" ausüben und sich besonders detaillierte Service Level Agreements (SLA's) zusichern lassen. Ob diese aus den Kombinationen Schenkung + Supportvertrag oder Kauf + Supportvertrag resultieren, erscheint eher zweitrangig. Bedeutend ist, dass die vereinbarte Leistung tatsächlich so erbracht werden kann, damit sie den hohen Anforderungen der öffentlichen Verwaltung entspricht.

Interessante Details hinsichtlich der Vertragsgestaltung liefern die AVB der Bundesbeschaffung GmbH[16].

5.4 (ÖFFENTLICH-)RECHTLICHE ÜBERLEGUNGEN ZU OPEN SOURCE

5.4.1 Staatsrechtliche Aspekte

Legalitätsprinzip

Der Erwerb von Software durch die öffentliche Hand zählt zu den Privatwirtschaftsverwaltungsagenden des Staates. Mit Privatwirtschaftsverwaltung ist die Beteiligung der Hoheitsträger am Wirtschaftsgeschehen, vor allem der Auftritt der letzteren als Nachfrager am Markt, gemeint, der Staat verhält sich also wie ein Unternehmen. Gem. Art. 18 Abs. 1 B-VG darf „die gesamte staatliche Verwaltung …nur aufgrund der Gesetze ausgeübt werden" (Legalitätsprinzip). Ob neben der Hoheitsverwaltung auch die Privatwirtschaftsverwaltung dem Legalitätsprinzip unterliegt, ist umstritten: die heute überwiegende Auffassung verneint dies ohne überzeugende Argumente.[17]

---

[13] siehe http://www.heise.de/newsticker/meldung/49735
[14] siehe http://www.oesterreichonline.at/report/artikel.asp?kid=2&mid=1&aid=2741
[15] z. B. mittels Download
[16] siehe unter https://bbg.portal.at/Internet/Dateien/AVB-IT Stand Sept._2003.pdf
[17] vgl. Walter/Mayer, Bundesverfassungsrecht RZ 570

In diesem Zusammenhang wurde auch die These des **„differenzierten Legalitätsprinzips“** entwickelt, nach welcher Art. 18 Abs. 1 B-VG für die Privatwirtschaftsverwaltung eine – im Vergleich zur Hoheitsverwaltung – weniger strenge Bindung an das Gesetz normiere. Selbstverständlich dürfen hier die vergaberechtlichen Bestimmungen nicht außer Acht gelassen werden, dazu jedoch später.

Eine allfällige Förderung von Open-Source-Produkten ist vom Einsatz letzterer zu unterscheiden und bedürfte einer gesetzlichen Determinante, z. B. in der Form eines „Selbstbindungsgesetzes“. Es muss freilich gesagt werden, dass eine solche Förderung sowohl rechtlich als auch politisch durchaus problematisch wäre. Einerseits könnten Grundrechte (Gleichbehandlung, Erwerbstätigkeit, Eigentum) berührt werden, andererseits wäre ein solcher Eingriff als besonders unliberal und daher als bedenklich zu bezeichnen.

Gleichheitssatz – E-Government für alle?

Die Gleichheitssatz-Problematik beim Einsatz von Open Source Software im Bereich des E-Governments stellt sich in zwei Hauptaspekten dar:

1. Werden die Hersteller proprietärer Software gegenüber den Herstellern von offener Software, falls letztere durch den Staat gefördert werden, diskriminiert?

Das Wesen des Gleichheitsgrundsatzes ist die Gleichheit vor dem Gesetz. Wie im Kapitel davor ausgeführt, könnte eine Förderung von Open-Source-Produkten durchwegs problematisch sein. Im Hinblick auf den Gleichheitsgrundsatz ist letztere, abzulehnen.

2. Werden die E-Government-Applikationen für alle User, gleich welchen Betriebssystems, zugängig sein?

Hier stehen zwei Optionen zur Wahl:

a) Alle Programme, die zur Nutzung der E-Government-Angebote notwendig sind, werden für die Software des Marktführers optimiert. Dies hätte folgende Konsequenzen:

   • Ein Teil der Bürger wird von der Nutzung dieser Anwendungen ausgeschlossen oder gezwungen, sich diese Software zu kaufen.

   • Bereits bestehende monopolartige Strukturen würden sich weiter verfestigen.

b) Die Anwendungen werden so konzipiert, dass keine Einschränkungen für die Clients resultieren. Annahmen über bestimmte Versionen von Betriebssystemen, Office-Programmen oder Browsern werden bei der Entwicklung nicht gemacht.

Die zweite Option erscheint hier eindeutig „gleichheitssatzfreundlicher“.

Vergaberecht

Der Begriff des Vergaberechts umfasst die Gesamtheit der Regeln und Vorschriften, die staatlichen oder staatsnahen Rechtsträger eine bestimmte Vorgehensweise beim Einkauf von Sachmitteln und Dienstleistungen vorschreiben.

Open Source und Vergaberecht

Das gesamte System des Vergaberechts ist von einigen grundlegenden Prinzipien durchzogen. Es soll daher untersucht werden, inwiefern sie mit dem Wesen von Open Source vereinbar sind.

Open Source und Wettbewerbsprinzip
Das Wettbewerbsprinzip ist eines der Kernprinzipien des Vergaberechts. § 16 Abs. 1 BVergG fordert „einen freien und lauteren Wettbewerb", das gesamte Vergabeverfahren hat diesen Grundsatz zu verwirklichen.

Im Rahmen der Beschaffung von Open-Source-Produkten stellt sich das Problem einer wettbewerbskonformen Vergabepraxis an zwei Stellen:

- Der öffentliche Auftraggeber kann bereits die Leistungsbeschreibung so abfassen, dass schon auf dieser Stufe eine unzulässige Verengung des Wettbewerbs erfolgt.

- Weiters stellt sich die Frage, ob bei einer Beschaffung von Software mit offenem Quellcode ohne Not auf einzelne Hersteller und/oder Distributoren Bezug genommen werden darf, indem der öffentliche Auftraggeber die Beschaffung beispielsweise eines „Linux-Betriebssystems" fordert[18] und damit sowohl Anbieter proprietärer als auch offener Software von der Ausschreibung ausschließt.

Open Source und Gleichbehandlungsprinzip
Der Gleichbehandlungsgrundsatz manifestiert sich im Diskriminierungsverbot (Verbot unmittelbarer wie mittelbarer Diskriminierungen). So stellen etwa „hersteller- und markenbezogene" Ausschreibungen in der Regel sowohl eine Verletzung des Wettbewerbsgrundsatzes als auch des Gleichbehandlungsgrundsatzes dar.

Open Source und Wirtschaftlichkeitsgrundsatz
Gem § 99 BVergG ist der Zuschlag auf das technisch und wirtschaftlich günstigste Angebot zu erteilen (Bestbieterprinzip). Erst in zweiter Linie stellt das Gesetz auf den billigsten Preis ab. Hauptziel dieser Regelung ist eine sparsame Verwendung öffentlicher Budgetmittel. Das Bestbieterprinzip könnte jedoch bei der Beschaffung von Open-Source-Produkten im Vergleich mit dem Einsatz proprietärer Software problematisch sein.

Setzt der Wirtschaftlichkeitsgrundsatz einen Preisvergleich voraus, müssen die Preise im Vergabeverfahren auch vergleichbar sein. Dieses Vergleichsproblem verschärft sich dadurch, dass

[18] So beim Bundestagsbeschluss, vgl Heckmann in Spindler „Rechtsfragen bei Open Source", S. 293

der Einsatz von Open-Source-Anwendungen sowohl im privaten wie auch im öffentlichen Bereich derzeit noch in den Anfängen steckt. Durch die neuesten Entwicklungen in München kann man zwar Vergleichswerte gewinnen, dafür wird aber auch eine entsprechende Dauer[19] vonnöten sein. Darüber hinaus unterscheiden sich die bestehenden Wirtschaftlichkeitsstudien hinsichtlich ihrer Beurteilungsmaßstäbe und sind sehr häufig interessengeleitet.

Open Source und Transparenzgrundsatz
Auch der Transparenzgrundsatz zählt zu den tragenden Säulen des Vergaberechts. Wesentliches Element dieses Grundsatzes sind die zahlreichen Publizitätserfordernisse im Rahmen einer Ausschreibung. Die Transparenz eines Vergabeverfahrens soll einen echten Wettbewerb ermöglichen, wobei auch hier die Vergabeunterlagen eine entscheidende Rolle spielen.

Insofern stellt sich an dieser Stelle ein weiteres Mal das Problem vergaberechtskonformer Produktspezifikation.

Open Source und vergabefremde Kriterien
Beispielhaft ist die Bedachtnahme auf die Umweltgerechtheit der Leistung und auf die Beschäftigung von Lehrlingen[20]. Es ist also davon auszugehen, dass Umwelt- und soziale Belange als Eignungs- und Zuschlagskriterien in ein Vergabeverfahren einfließen können. Ob letztere auch zu einer Privilegierung von Open Source führen könnten, ist derzeit noch fraglich.

Open Source als Ausschreibungsgegenstand am Beispiel des BVergG

Das Vergaberecht bindet den öffentlichen Auftraggeber bei der Beschaffung seiner Waren und Dienstleistungen im erheblichen Umfang. Damit unterscheidet sich die öffentliche Auftragsvergabe von dem privatautonomen Beschaffungswesen nicht öffentlicher Auftraggeber. Verstößt ein öffentlicher Auftraggeber gegen formelle und/oder materielle Regeln des Vergaberechts, so entsteht hier ein kaum zu kalkulierendes Verschleppungs- und damit Kostenrisiko.

Die Leistungsbeschreibung

Die Regelungen zur Leistungsbeschreibung in den §§ 74 ff. BVergG sind von grundlegender Bedeutung für das Vergabeverfahren. Die Leistungsbeschreibung hat dabei die Aufgabe, den Bedarf des Auftraggebers so genau wie möglich zu beschreiben, ohne aber auf der anderen Seite den möglichen Kreis der Bieter einzuschränken.

Hinsichtlich der Leistungsbeschreibungsart ist zu sagen, dass das BVergG die Funktionalität[21] in den Vordergrund stellt. Dem Bieter bleibt es bei einer funktionalen Leistungsbeschreibung überlassen, auf welche Art und Weise er die festgelegten Ziele erreichen will. Die Ausschreibung ist offener und damit wird auch der Bieterkreis größer.

[19] Das LiMux-Projekt läuft erst seit 2003
[20] § 16 Abs. 7 BVergG
[21] vgl § 74 Abs. 2 BVergG

Es verbleibt die Frage, welche Grundsätze bei der Gestaltung einer funktionalen Leistungsbeschreibung zu beachten sind. Bereits § 74 Abs. 1 BVergG gibt den Grundsatz, dass die Leistungen bzw. die Aufgabenstellungen **neutral** auszuschreiben sind. Gleichzeitig geht der § 74 Abs. 2 von einer Präzisierung der Beschreibung, die die Vergleichbarkeit der daraus entstandenen Angebote gewährleisten soll.

Probleme können sich aus dem § 74 Abs. 3 ergeben, die es dem Auftraggeber grundsätzlich verbietet, die Leistungsbeschreibung so abzufassen (zu präzisieren), dass bestimmte Bieter von vornherein Wettbewerbsvorteile genießen. Hinsichtlich dieses Verbotes wird dem öffentlichen Auftraggeber jedoch ein Spielraum eingeräumt, und zwar in den Vorschriften des § 75 Abs 8 und 9, die nur dann eine Wettbewerbsverletzung durch namentliche Nennung von Erzeugnissen oder Verfahren statuieren, soweit nicht besondere Umstände wie die Wahrung der technischen Einheit bei der Erweiterung oder Instandhaltung von Systemen dies notwendig macht. Demnach wird die Markenbezeichnung „Microsoft Word“ erlaubt sein, wenn die Behörde innerhalb einer bestehenden Softwareumgebung das Textverarbeitungsprogramm upgraden will. Nicht erlaubt wird die Beschränkung auf Open-Source-Technologie oder proprietäre Software sein, wenn ein Verwaltungsbereich technisch komplett neu ausgestaltet werden soll. Interessant ist auch die Frage, inwieweit (aus wirtschaftlichen Erwägungen) an vorhandene, verwertbare EDV-Infrastrukturen angedockt werden kann.

Primär kann die Antwort von der „Verwertbarkeit“ abhängen. Hieraus lässt sich also folgendes Zwischenergebnis folgern: Auf der Ebene der Leistungsbeschreibung darf der öffentliche Auftraggeber weder bestimmte Marken oder Hersteller noch eine Offenlegung des Quellcodes als Zuschlagskriterium vorschreiben. Vielmehr sind auch Ausschreibungen, die sich aus der Sicht des Auftraggebers subjektiv in erster Linie auf Open-Source-Produkte beziehen, im Sinne eines möglichst breiten Wettbewerbs in jeder Hinsicht produkt- und herstellerneutral auszuschreiben. Damit wird gewährleistet, dass sowohl Open-Source-Produkte wie auch proprietäre Software in die Angebotsprüfung gelangen können.

Angebotsprüfung und Zuschlagserteilung

Gemäß § 99 BVergG ist der Zuschlag dem technisch und wirtschaftlich günstigsten Angebot oder dem Angebot mit dem niedrigsten Preis zu erteilen. Die sich hieraus auf den ersten Blick ergebende freie Wahlmöglichkeit zwischen beiden Zuschlagsprinzipien wird jedoch insofern eingeschränkt, als eine Vergabe ausschließlich auf der Grundlage des Angebotspreises nur dann zulässig sein soll, wenn der Qualitätsstandard der Leistung in der Bekanntmachung oder in den Ausschreibungsunterlagen klar und eindeutig definiert ist, sodass die Festlegungen in der Ausschreibung qualitativ gleichwertige Angebote sicherstellen.[22] Dies ist vor allem unter dem Gesichtspunkt des „niedrigen“ Preises von Open-Source-Produkten, da keine Lizenzkosten anfallen, als besonders wichtig hervorzuheben. Der Zuschlag erfolgt sodann auf das wirtschaftlichste Angebot, der niedrigere Preis ist nur bei Gleichwertigkeit ausschlaggebend.

[22] § 67 Abs. 3 BVergG

5.6 ZUSAMMENFASSUNG

E-Government ist das Schlagwort der letzten Jahre. Die öffentliche Verwaltung ist offenbar auf dem besten Wege, die politischen Vorgaben, „alle Behördenwege zu elektronisieren, die sich elektronisieren lassen“, zu verwirklichen. Es ist ganz klar, dass die Wahl der Software dabei eine sehr wichtige Rolle spielen wird. Die Überlegungen, ob Open Source oder proprietäre Software zum Einsatz kommen soll, sind in vollem Gange. Für viele ist Open Source das Synonym für Freiheit und Offenheit: die vorliegende Studie konfrontiert diese Sichtweise mit den rechtlichen Rahmenbedingungen im öffentlichen Bereich.

Es steht fest, dass die rechtliche Grundlage des Open-Source-Vertriebs- und Lizenzmodels, das in den USA entwickelt wurde, auf Europa und speziell auf Österreich nicht eins zu eins übertragen werden kann. Gleichzeitig wirft der beabsichtigte Einsatz von Open-Source-Produkten im E-Government einige grundlegende Fragen auf. Vor allem im Sicherheitsbereich und im mit ihm eng verwandten Haftungsbereich gilt es ganz vorsichtig und behutsam vorzugehen, um eine gut funktionierende und zuverlässige Lösung auf die Beine zu stellen.

Nicht außer Acht darf die Grundrechtsproblematik bleiben. Die besprochene Lösung darf weder die User noch die Anbieter von Software, gleich welchen Vertriebmodells, benachteiligen.

Wie schon festgestellt, unterliegt die Anschaffung von Software für die gesamte öffentliche Verwaltung den Vergaberechtsbestimmungen. Um Kosten- und Verschleppungsrisiken aus dem Weg zu gehen, ist die Verwaltung gut beraten, rechtssichere Vergabeunterlagen auszuarbeiten. Dabei ist es von Bedeutung, dass keiner der Anbieter diskriminiert wird, und dass die Angebote auf ihre Vergleichbarkeit hin geprüft werden. Diese Wertungsproblematik verschärft sich im Zusammenhang mit der Beschaffung von Open-Source-Produkten dadurch, dass die Idee, der öffentliche Auftraggeber erhalte im Rahmen seiner Beschaffungsprozesse „etwas kostenlos“[23], dem Vergaberecht grundsätzlich fremd ist, mithin also die Idee von Open-Source-Produkten grundsätzlich einen vergaberechtlichen Fremdkörper darstellt.

Alles in allem kann zwar nicht gesagt werden, dass Open Source Software für den öffentlichen Bereich ungeeignet ist. Aus juristischer Sicht muss dennoch betont werden, dass zahlreiche Rechtsfragen, die mit einem allfälligen Einsatz dieser Software in der Verwaltung einhergehen, noch offen sind und sohin eine gewisse Rechtsunsicherheit besteht.

[23] Bezogen auf die „Anschaffung“ der Software an sich.

Österreichischer Städtebund
1082 Wien, Rathaus
Telefon: 01/4000-89980
Telefax: 01/4000-7135
E-Mail: post@stb.or.at
Internet: http://www.staedtebund.at