Stärken und Schwächen freier und Open-Source-Software im Unternehmen

THOMAS WIELAND

Zusammenfassung

Freie und Open-Source-Software stößt bei einer wachsenden Zahl von Unternehmen auf Interesse, wobei die meisten lediglich an eine Nutzung bestehender Software denken, manche aber auch eigene Entwicklungen freigeben wollen. Damit die Entscheidung für den Einsatz von bzw. das Engagement für freie und Open-Source-Software nicht nur „aus dem Bauch heraus" getroffen werden muss, ist eine sorgfältige Analyse der individuellen Situation der Firma notwendig. Diese setzt natürlich voraus, dass nicht nur die Besonderheiten dieser Art von Software bekannt sind, sondern auch deren Stärken und Schwächen, Potenziale und Bedrohungen. In diesem Beitrag soll ein knapper Überblick über die wichtigsten Aspekte aus diesen vier Kategorien gegeben werden, um den unternehmerischen Entscheidungsprozess zu erleichtern und einige Hilfestellungen zur Auswahl zu geben. Dabei stehen naturgemäß hauptsächlich strategische und wirtschaftliche Gesichtspunkte im Vordergrund; die unbestreitbaren technischen Vorteile werden an anderer Stelle in diesem Band bereits ausführlich gewürdigt.

Einleitung

Bei immer mehr Firmen und Behörden wächst das Interesse an freier und Open-Source-Software (im Folgenden kurz OSS). Zunächst nutzen sie sie lediglich, wie sie bisher auch proprietäre Software genutzt haben, beispielsweise aus Gründen der Kostensenkung. Gerade Entwickler in Industrie- oder Dienstleistungsunternehmen stellen oft jedoch mit der Zeit fest, dass die eine oder andere OSS-Bibliothek oder -Anwendung eine ausgezeichnete Grundlage für die eigene Produktentwicklung darstellen würde. Damit wird die Verflechtung mit der so genannten „Open-Source-Szene" immer enger und natürlich auch komplizierter. Schließlich kommt auch eine wachsende Zahl von Unternehmen zu dem Schluss, dass sie eigene Software als Open Source freigeben wollen.

Die Diskussion um Open Source wird mittlerweile in einer so breiten Öffentlichkeit geführt, dass so ziemlich jeder Verantwortliche in Unternehmen, die Informationstechnologien einsetzen, schon einmal mit dem Thema in Kontakt kam. Die Gefühlslage kann dabei schnell von einer Ablehnung aus einer vagen Unsicherheit heraus bis hin zu Begeisterung für die Idee schwanken. Der derzeitige allgegenwärtige Hype für OSS trägt noch ein Übriges dazu bei. Doch unternehmerische Entscheidungen sollten nicht aus unbestimmbaren Gefühlen heraus getroffen werden – und auch nicht nur als Nachahmung eines Trends. Jeder Verantwortliche sollte an-

hand einer Reihe von konkreten Argumenten für seinen Einzelfall entscheiden, ob der Einsatz von bzw. das Engagement für freie und Open-Source-Software in seiner Firma sinnvoll ist oder nicht.

Eine klassische Technik, die für die Untersuchung solcher Entscheidungssituationen gerne verwendet wird, ist die so genannte SWOT-Analyse. Diese vier Buchstaben stehen dabei für die englischen Begriffe strengths, weaknesses, opportunities und threats – also Stärken, Schwächen, Potenziale und Bedrohungen. Nach diesem Schema wollen auch wir hier untersuchen, welche Argumente für und gegen freie und Open-Source-Software im Unternehmen sprechen. Dabei soll sowohl der reine Nutzungsfall als auch der des Mitwirkenden an OSS betrachtet werden.

In diesem Beitrag wollen wir eine Software als „Open-Source-Software" bezeichnen, wenn die Lizenz für ihre Verbreitung und Nutzung mit der Open-Source-Definition in Einklang steht, die von der Open-Source-Initiative (2003) veröffentlich wurde. Diese Definition umfasst zehn klare und strenge Regeln, die eine Software und ihre Lizenz erfüllen müssen, damit die Bezeichnung „Open Source" gerechtfertigt ist. Bereits die Lizenzmodelle, die von dieser Definition umfasst werden, sind äußerst vielfältig. Ein noch viel breiteres Spektrum machen aber die Projekte selbst aus. Allein beim Mediator SourceForge[1], einem Portal für OSS-Projekte, waren Ende 2003 über 70.000 verschiedenen Projekte registriert. Diese unterschieden sich sehr stark in Größe, d. h. Zahl der Mitwirkenden bzw. Umfang der entwickelten Software, und Status, von vagen Ideen über viel versprechende Anfänge bis zu reifen Projekten.

Daraus ist ersichtlich, dass alle Aussagen, die im Folgenden zu treffen sind, immer nur für einen Teil von Projekten Gültigkeit besitzen. Für jedes Argument für oder gegen OSS existieren sicherlich mehrere Gegenbeispiele, die genau diesen Gesichtspunkt widerlegen. Es muss sich daher also nicht um eine Damm öffnende Grundsatzentscheidung handeln, wenn ein Unternehmen sich zum Einsatz einer bestimmten Open-Source-Software entschließt. Es muss nach wie vor in jedem Einzelfall abgewogen werden, welcher Schritt aus unternehmerischer Sicht sinnvoller ist.

Genauso wenig, wie es „das Open-Source-Projekt" gibt, gibt es „die Open-Source-Community". Eine der verbreiteten grundlegenden Fehlannahmen ist nämlich, die Community als eine einzige einheitliche Menge von Menschen zu betrachten. Zu jedem Projekt bildet sich vielmehr eine eigene Community heraus. Diese kann sich mit der anderer Projekte überlappen, wenn sich Entwickler etwa in mehreren Projekten engagieren, ist aber mit den meisten völlig disjunkt. Sie kann nur aus einer Person bestehen oder aber aus vielen Tausenden.

Für unsere Zwecke hier definieren wir also:

- Eine „Projekt-Community" ist ein virtuelles Team, das sich zeitweise auf freiwilliger Basis zusammensetzt, um eine bestimmte Aufgabe gemeinsam zu lösen.
- Die „Community" ist die Gesamtheit aller Projekt-Communities.

[1] SourceForge: http://sourceforge.net.

Open-Source-Jahrbuch 2004, hrsg. von Robert A. Gehring und Bernd Lutterbeck, Berlin (Lehmanns Media) 2004.

Zu beachten ist, dass die Grenzen hier fließend sind, abhängig von den Rollen, die ein Individuum in Bezug auf das jeweilige Projekt spielt. Ist eine Person, die die Software unverändert nutzt, bereits Mitglied der Community? Nach obiger Definition nicht. Aber ist jemand, der die Software an seine Bedürfnisse anpasst, ein Mitglied? Schon eher, wenn man den Begriff der „Aufgabe" weit genug fasst.

Für das Weitere wollen wir den Begriff der Community in der Tat recht weit fassen. Es geht darum, nicht nur alle Entwickler zu erfassen und zu erreichen, die bereits einen Beitrag leisten, sondern das Forum hervorzuheben, in dem Nutzer und Entwickler sowie alle, die potenzielle Beiträge zur Verfügung haben, sich austauschen, Probleme diskutieren, Anregungen und Vorschläge machen sowie Lob und Kritik aussprechen können. Eben in dieser Art von Forum unterscheidet sich OSS in der Tat grundsätzlich von klassischer proprietärer Software.

Stärken freier Software

Open-Source-Software würde nicht auf so viel Interesse stoßen, wenn sie nicht einige unbestreitbare Stärken vorzuweisen hätte. Diese wollen wir zunächst betrachten.

Einer der renommiertesten Vorreiter der Open-Source-Bewegung ist Eric Raymond, der mit seinen Essays „The Cathedral and the Bazaar" und „The Magic Cauldron" (beide abgedruckt in Raymond 1999) viele wertvolle Argumente zur Diskussion beigetragen hat. Er sieht einen der wichtigsten Vorteile von Open Source im „peer review"-Prinzip. Das bedeutet, dass Programme anhand ihres Codes von anderen Experten beurteilt werden können. Durch diese kritische Überprüfung können alle Arten von Fehlern relativ rasch aufgedeckt und beseitigt werden. Denn die Wahrscheinlichkeit, dass ein Fehler unbemerkt bleibt, ist bei mehreren Hundert Programmierern um Größenordnungen geringer als bei ein paar wenigen – sofern ein „code review" bei der geschlossenen Entwicklung überhaupt stattfindet. Aus diesem Grund ist das Open-Source-Modell besonders für Infrastrukturkomponenten wie Betriebssystemkernels, Treiber oder Netzwerkdienste sinnvoll.

Nicht nur für die Qualität ist das „peer review" vorteilhaft, auch die Sicherheit profitiert davon. Denn eine wirklich ernsthafte Sicherheitsüberprüfung muss sich auch auf den Code beziehen; nur Algorithmen und Implementierungen, die von verschiedenen unabhängigen Seiten als sicher eingestuft werden, kann letztendlich vertraut werden. Aus diesem Grund setzen Organisationen, bei denen Sicherheit eine große Rolle spielt, wie der amerikanische Geheimdienst NSA, das deutsche Bundesamt für Sicherheit in der Informationstechnik sowie das Auswärtige Amt immer häufiger von ihnen selbst als sicher zertifizierte OSS-Systeme ein.

Zudem folgt die Gesamtkonzeption bei freier Software meist einem strengeren Sicherheitsmodell. Insbesondere das Betriebssystem GNU/Linux mit seinen verschiedenen Infrastrukturkomponenten auf Open-Source-Basis erlaubt eine scharfe Trennung zwischen Rechten des Anwenders und des Administrators. Hierzu kommt beispielsweise das Bundesinnenministerium in einer Studie (Stör, 2003) zu dem Schluss: „Beim Systemmanagement folgen die OSS-Betriebssyteme ihrer Herkunft entsprechend dem UNIX-Weg. […] Für Administratoren und Betriebsorgani-

sation sind bei einer Migration auch konzeptuelle Änderungen zu erwarten, die insbesondere bezüglich der Sicherheit große Fortschritte ermöglichen. Die Sicherheit und Zuverlässigkeit, die den Linux-Systemen allgemein zugeschrieben wird, ist nicht zuletzt das Resultat des Systemmanagement."

Die Offenheit ist aber auch in anderer Hinsicht eine Stärke. Wenn der Code einer Software offen liegt, kann jeder sich die darin verfolgte Problemlösung ansehen und daraus lernen. Eine bereits vorhandene und bewährte Lösung kann dann immer wieder verwendet werden. Der Einzelne muss sich nicht jedes Werkzeug selbst herstellen, sondern kann dieses einsetzen, um darauf eine komplexere Lösung zu erstellen. Dieses allgemeine Innovationsprinzip, das die technische Entwicklung der Menschheit charakterisierte, ist mit geschlossener Software kaum möglich.

Darüber hinaus hat sich gezeigt, dass einmal entdeckte Fehler und Sicherheitsprobleme bei OSS im Durchschnitt etwa sechsmal schneller behoben werden als bei gleichartiger proprietärer Software (LW 2003). So zeigte etwa eWeek (2002) anhand einer Reihe von Beispielen, dass Sicherheitsprobleme in OSS-Projekten sehr ernst genommen und äußerst schnell behoben werden, während Hersteller kommerzieller Software sich oft zunächst um eine Verharmlosung des Problems bemühen.

Auch ist es bei OSS normalerweise nicht nötig, die gesamte Anwendung zu aktualisieren, nur um punktuelle Fehlfunktionen zu beheben. Der Benutzer kann also bei Problemen stets auf eine rasche und zielgerichtete Hilfe hoffen. Bei größeren und sehr aktiven Projekten reagieren die Entwickler im Allgemeinen auf E-Mail-Anfragen aller Art innerhalb weniger Stunden mit einer konstruktiven Antwort.

Schwächen freier Software

Natürlich ist aber auch freie und Open-Source-Software nicht immer ohne Probleme. Da hinter einer Open-Source-Software nur selten eine Firma steht, darf man nicht erwarten, eine solche Anwendung in genau derselben Ausstattung zu bekommen wie eine kommerzielle – obschon selbst das zuweilen der Fall ist. Die meisten Entwickler sind eher an Funktionalität als an Ergonomie oder umfassender Dokumentation interessiert. Insofern kann man Open-Source-Applikationen als „Rohprodukte" begreifen, die erst für den täglichen Gebrauch nutzbar gemacht werden müssen.

Das kann durch den Anwender selbst geschehen, wenn er über das nötige Know-How verfügt oder sich die Kenntnisse im Internet zusammensuchen möchte, oder über Dienstleister, sowohl firmeninterne als auch -externe. Eine Reihe von Unternehmen haben sich als Service-Anbieter für Open Source einen Namen gemacht, darunter natürlich auch die großen Linux-Distributoren Red Hat und SuSE. Wer also ernsthaft Open-Source-Anwendungen einsetzen will, sollte einen Support-Vertrag dazu abschließen, um die optimale Verfügbarkeit für alle Geschäftsfälle gewährleisten zu können. Damit ist nur vordergründig eine neue Abhängigkeit zu einem „Hersteller" geschaffen. Denn da die Quellen offen sind, kann sich jeder Dienstleister das nötige Wissen aneignen, und der Auftraggeber kann ihn – theoretisch – jederzeit austauschen.

Auch die Softwarepakete selbst haben manchmal noch nicht den Grad der Reife erreicht, damit sie für den Alltagseinsatz im Unternehmen geeignet sind. Die Schwierigkeit für den potenziellen Nutzer besteht hier bereits darin, den Reifegrad überhaupt zu erkennen. Versionsnummern allein sagen dazu kaum etwas aus; eine Version 0.4 einer Software kann ein stabileres Programm bedeuten als eine 1.1 einer anderen. Zum Glück setzen sich auch in Open-Source-Projekten immer mehr etablierte Techniken des „software engineering" durch, etwa die des „release management". Wird ein Entwicklungsstand beispielsweise als „stable release" bezeichnet, kann man im Allgemeinen von einer stabilen Version ausgehen. „Unstable" oder „snapshots" ist dagegen eher mit Vorsicht zu begegnen. Der hohe Anspruch vieler OSS-Entwickler führt jedoch ab und an dazu, dass selbst ausgereifte Programme noch als „unstable" klassifiziert werden, die bei einem kommerziellen Entwicklungszyklus schon lange auf dem Markt wären.

Der erste Schritt bei der Auswahl eines OSS-Produkts für den Firmeneinsatz ist also die umfassende Evaluation, die gegebenenfalls auch ein Dienstleister durchführen kann. In dieser sollte durch eingehende Tests festgestellt werden, ob die Software in der aktuellen Version die in sie gesetzten Erwartungen auch erfüllen kann.

Natürlich muss man sich vorher schon sicher sein, das richtige Programm ausgewählt zu haben. Denn auch das ist eine „Schwäche" von OSS – wenn man so will. Für fast alle Aufgaben gibt es mehr als ein Projekt, das eine Lösung dazu erarbeiten will. Der Ansatz kann dabei ebenso unterschiedlich sein wie der Umfang und der Reifegrad. Auf den ersten Blick kann man daher kaum erkennen, welche Software nun besser geeignet ist. Kriterien für die Auswahl können u.a. sein:

- Zugesicherte oder getestete Funktionalität
- Stabilität
- Qualität von Architektur und Design
- Lizenzmodell
- Unterstützung durch die Projekt-Community
- Zahl der Mitarbeiter und Dauer des Projekts
- Engagement von Firmen.

Ein Fallbeispiel für die Auswahl eines OSS-Pakets wurde etwa von Hang et al. (2004) beschrieben.

Ein weiteres immer wieder auftauchendes Problem bei freier Software ist die gegenseitige Abhängigkeit. Viele Entwickler versuchen, sich mit den Programmierern anderer OSS-Projekte zu koordinieren, um so aufeinander aufbauen zu können. Das führt für den Anwender dazu, dass die Version, die er ausgewählt hat oder die er auch nur ausprobieren möchte, auf seinem Zielsystem nicht läuft, da sie von einer anderen Software abhängt, die augenblicklich nicht installiert ist. Erschwerend kommt leider häufig dazu, dass die nun für die Installation benötigten Programme zwar vorhanden sind, aber in einer älteren Version. Um ein Update für diese einzuspielen, brauchen sie wiederum aber Updates von weiterer Software, von der sie selbst abhängen. Das kann sich über mehrere Stufen fortsetzen, bis der Anwender feststellt, dass er ein vollständiges Upgrade seiner Betriebssystemplattform bräuchte, um ein bestimmtes OSS-Paket überhaupt starten zu können.

Hierbei hilft nur, sich frühzeitig darüber klar zu werden, welche Programme in welchen Versionen geschäftskritisch sind und wie weit die Bereitschaft zum Update geht. Sind die Erfordernisse, die eine bestimmte Software stellt, zu umfangreich, muss auf die Evaluation beziehungsweise Nutzung dieser Software verzichtet werden – sofern nicht in Absprache mit den Entwicklern eine Zwischenlösung gefunden werden kann.

Potenziale freier Software

Was bedeuten diese Eigenschaften von freier und Open-Source-Software nun für den Einsatz im Unternehmen? Welche Potenziale können damit erschlossen werden, und welche strategischen Vorteile ergeben sich daraus? Mit diesen Fragen wollen wir uns nun beschäftigen.

Einer der sicher auffälligsten Vorteile von freier Software sind die geringen Kosten. Dabei ist „frei" im Grunde im Sinne von „freier Rede" und nicht von „Freibier" zu verstehen. Bei freier Software hat der Benutzer das Recht, sie zu kopieren, zu verbreiten, zu analysieren und zu verbessern. Die unbedingte Voraussetzung dafür ist, dass der Anwender Zugang zum Quellcode der Software erhält (Free Software Foundation, 1999). Es ist nicht einmal untersagt, dass ein Anbieter Open-Source-Software gegen Entgelt verkauft. Nur darf er keinem anderen verbieten, die gleiche Software kostenlos abzugeben. Daher hat es sich heute eingebürgert, dass nur für die Bereitstellung, die Aufarbeitung und den Datenträger Gebühren verlangt werden. Nichtsdestotrotz kann man alle freie und Open-Source-Software kostenfrei als Download über das Internet erhalten – von den Gebühren für den Internetzugang einmal abgesehen.

In der Praxis kann man also die Anschaffungskosten für OSS-Programme vernachlässigen. Dabei spielt es nicht einmal eine Rolle, ob nun eine Kopie des Programms (etwa auf einem Server) oder Dutzende bis Tausende (auf Arbeitsplatzrechnern) eingesetzt werden sollen. Dies stellt einen fundamentalen Unterschied zu kommerzieller Software dar, bei der ja pro genutzter Lizenz Gebühren anfallen.

Jeder erfahrene IT-Verantwortliche weiß indessen, dass die Lizenzkosten einer Software nur einen geringen Teil der Gesamtkosten ausmachen, die mit dieser Software verbunden sind. Dazu gehören auch noch die Kosten für die erforderlichen Hardwareressourcen, für Systemadministration und Support, für Datenkommunikation und Netzwerk, für benutzerbezogene und technische Ausfallzeiten sowie für den Aufbau oder Einkauf des nötigen Know-How. Diese Summe der gesamten Kosten eines Investionsgutes (hier also einer Software) bezeichnet man auch als „Total Cost of Ownership" (TCO).

Es gibt eine ganze Reihe von Studien, in der die TCO von proprietärer und freier Software miteinander verglichen werden. Die meisten davon stellen GNU/Linux und Windows oder ein kommerzielles Unix gegenüber. So kommen beispielsweise Gillen et al. (2001) zu dem Schluss, dass GNU/Linux im Bereich Intranet/Internet nur 55% der TCO-Kosten pro Jahr verursacht, die ein vergleichbares System auf RISC/Unix-Basis mit sich bringen würde. Das australische Marktanalyse-Unternehmen Cybersource (2002) stellt bei einem ähnlichen Vergleich von GNU/Linux mit

Windows eine Kostenersparnis von bis zu 34,3 % über einen Dreijahreszeitraum fest. Der Anwalt Brendan Scott (2002) hat mit eher theoretischen Überlegungen gezeigt, warum freie Software langfristig niedrigere Gesamtkosten haben muss. Und sogar der CEO von Sun Microsystems, Scott McNealy, spricht im Zusammenhang mit GNU/Linux immer wieder von bis zu 90% niedrigeren IT-Kosten (zitiert nach Nowak 2003).

Was heißt das Ganze nun für die Praxis? Hier haben Binder und Lipski (2003) recht klar herausgestellt: Es gibt ein beachtliches Einsparpotenzial durch OSS – davon profitieren allerdings große Unternehmen in sehr viel stärkerem Maße als kleine, da jene oftmals schon internes Open-Source-Know-How haben und es sich nicht teuer extern einkaufen müssen. Außerdem fallen bei großen Organisationen Lizenzkosten proprietärer Software für Tausende von Arbeitsplätzen stärker ins Gewicht. Somit sind pauschale Modellrechnungen immer nur Anhaltspunkte, die für das einzelne Unternehmen lediglich begrenzte Aussagekraft haben. Jede Firma muss daher selbst analysieren, welchen Bedarf an IT-Unterstützung sie hat und welche Kosten dafür proprietäre und OSS-Lösungen mit sich bringen.

Neben reinen Kostenargumenten gibt es aber auch eine Reihe von strategischen Aspekten, die für freie und Open-Source-Software sprechen. Baut etwa ein Benutzer einen wichtigen Teil seines Geschäftes auf der Software eines bestimmten Herstellers auf, so macht er sich von diesem abhängig. Sind etwa in der nächsten Version größere Änderungen enthalten, muss der Anwender die entsprechenden Anpassungen seiner Umgebung nachvollziehen, wenn er nicht seine bisherigen Investitionen verlieren will. Er ist also gezwungen, synchron mit dem „Innovationszyklus" des Herstellers zu bleiben, auch wenn er eigentlich keinen Bedarf an Aktualisierungen bzw. Neuerungen hat. Selbst wenn nur die Unterstützung für ein eingesetztes Produkt wegfällt, ist der Anwender zu einem Update veranlasst. Ein Beispiel ist die gegenwärtige Welle der Plattformwechsel bei Arbeitsplatzsystemen als Folge der Ankündigung der Einstellung des Supports für Windows NT 4 durch Microsoft.

Besonders kann es kritisch werden, wenn der Hersteller plötzlich nicht mehr am Markt ist – weil er in Konkurs gegangen ist, aufgekauft wurde usw. Hier müssen die Anwender um die Weiterexistenz ihrer Software fürchten, was beim Einsatz in kritischen Bereichen besonders ärgerlich ist.

Beide Aspekte fallen bei Open Source weg. Da es keinen direkten Hersteller gibt (höchstens einen „Projektleiter"), begibt sich der Anwender in keinerlei Abhängigkeiten. Zur Not kann er die Software selbst weiter pflegen, wenn sie für ihn einen besonderen Stellenwert hat.

Dabei haben wir bislang nur die Potenziale von OSS für reine Anwender betrachtet. Zum Abschluss dieses Abschnitts wollen wir noch untersuchen, welche Gründe aus Sicht eines Softwareherstellers dafür sprechen, ein Programm als Open Source zu veröffentlichen (siehe auch Wieland 2001).

Früher entstanden die meisten Open-Source-Projekte dadurch, dass ein Programmierer eine Idee verwirklichen wollte oder ein bestimmtes Tool benötigte, das er selbst schreiben musste, dann aber mit anderen teilen wollte. Wenn eine Firma heute ein solches Projekt ins Leben ruft, geschieht dies kaum noch aus technischen, sondern meist aus Marketingüberlegungen. Auf diese Weise kann sich das Unter-

nehmen nicht nur als offen und innovativ hervorheben, sondern auch bestimmte Benutzergruppen an sich binden oder sich in neuen Märkten positionieren.

Außerdem können durch Open-Source-Programme eigene Standards mit viel größerer Breitenwirkung und in kürzeren Zeiträumen auf dem Markt etabliert werden, als dies mit kostenpflichtiger Software möglich wäre. Diese Strategie kann man beispielsweise bei Sun mit Java und bei IBM mit Internet- und XML-Software beobachten. Letztlich versichern zwar alle Hersteller, keine proprietären Standards zu wollen, sondern sich an offene Standards der internationalen Gremien (IETF, W3C etc.) zu halten. Da die Verabschiedung von Standards durch diese jedoch meist recht lange dauert, kann der Hersteller seine eigenen Vorstellungen am Markt und bei der Standardisierung am besten durchsetzen, der über die größte Anwenderbasis verfügt.

Damit ist nämlich auch gleich das Feld für die nächste Stufe bereitet: Der Hersteller findet bei genügender Akzeptanz des von ihm favorisierten Standards einen lohnenden Markt für kommerzielle Produkte vor, die darauf aufbauen und die Möglichkeiten der OSS-Programme ergänzen beziehungsweise erweitern. Das können größere, kostspielige Softwarepakete, etwa für die Entwicklung oder Administration, sein genauso wie Hardware. Auch dafür ist IBM mit der WebSphere-Produktreihe, die auf Java- und Internetstandards sowie auf Eclipse setzt, oder der Linux-Portierung auf AS/400 ein gutes Beispiel.

Für die Anwender ist es mit der kostenlosen Software allein oft nicht getan. Je komplexer die Anwendung, desto mehr Bedarf besteht an Unterstützung bei der Anpassung an die Umgebung des Benutzers und an der Erarbeitung maßgeschneiderter Lösungen. Obwohl dies die klassische Einnahmequelle einer Open-Source-Firma ist, ist diese Strategie nach wie vor lohnenswert.

Die Entwicklungsleistung kann dabei auch teilweise von einzelnen Anwendern kommen. Für manche Kunden kann das Programm so wichtig sein, dass sie eigene Ressourcen für dessen Pflege und Ausbau bereitstellen – Erweiterungen, die sie auf Grund des Lizenzmodells meist wieder an den Hersteller zurückmelden müssen. Auf diese Weise können also Entwicklungskosten vom ursprünglichen Produzenten auf seine Kunden verlagert werden – in vielen Fällen ein nicht unwesentliches Argument.

Aber nicht nur die Entwicklung kostet Geld; oft ist der laufende Unterhalt für die Pflege einer Software sogar noch teuer. Viele Firmen pflegen Softwareprodukte nicht, weil sie mit dem laufenden Lizenzverkauf noch etwas verdienen, sondern nur, um bestehende Kunden zu bedienen. In solchen Fällen bietet sich die Freigabe der Software als Open Source an, da man den Pflegeaufwand damit zumindest teilweise auf andere verteilen kann. Dazu ist allerdings auch der vorbereitende Aufbau einer entsprechenden Community nötig . Ein prominentes Beispiel für diesen Schritt ist AOL/Netscape mit der Freigabe des Mozilla-Browsers – auch wenn dabei im Laufe der Zeit noch ein paar andere Einflüsse hinzukamen.

Schließlich ist noch zu bedenken, dass auch in einem Unternehmen hinter einer Software nicht immer riesige Teams stehen, sondern oft nur zwei oder drei Mitarbeiter. Und von diesen hängt die Software dann auch ab. Selbst wenn die Programme ausreichend dokumentiert sind, lassen sie sich im Allgemeinen nicht ohne Wei-

teres durch andere übernehmen. Falls also einer der Mitarbeiter die Firma verläßt oder eine neue Aufgabe übernimmt (oder gar alle), ist die Gefahr groß, dass die Anwendung nicht gepflegt wird, hinter den sonstigen technischen Entwicklungen hinterherhinkt und langsam unbrauchbar wird. Durch die Veröffentlichung der Software als Open Source kann man dieses Problem vermeiden, denn man übergibt das Programm einer – hoffentlich – größeren Gemeinschaft, die sich um den Fortbestand (und somit um die Sicherung der ursprünglichen Investition in die Entwicklung) kümmern wird. So werden vielleicht nicht unmittelbar Kosten eingespart, dafür aber das Risiko der Unbrauchbarkeit auf eine breite Anwenderbasis außerhalb der eigenen Firma verteilt und die Abhängigkeit von einzelnen Entwicklern reduziert.

Bedrohungen für freie Software

Aber selbst freie und Open-Source-Software und deren Anwender sehen sich einigen Bedrohungen gegenüber. Zunächst bleibt festzuhalten, dass die Grundidee von freier Software nicht nur den freien Zugang beinhaltete. Man stellte sich vielmehr vor, dass die unentgeltliche Software, die man von anderen erhält, durch eigene Mitarbeit an diesem oder einem anderen Projekt kompensiert wird. Auf diese Weise haben alle einen Nutzen, wobei ihr eigener Aufwand vertretbar bleibt.

Dieses Prinzip ist heute offenbar durchbrochen. Es gibt wesentlich mehr Nutzer freier Software als Entwickler. Und die Nutzer sehen darin längst nicht mehr eine moralische Verpflichtung zur Gegenleistung. Solange es trotzdem genügend Entwickler gibt, ist dagegen auch nichts einzuwenden. Denn die wachsende Verbreitung von OSS allein ist schon ein Erfolg, der viele zusätzliche positive Veränderungen mit sich bringt. Zudem haben sich die Motivationen der Teilnehmer an OSS-Projekten deutlich gewandelt. Wie Hars und Ou (2001) sowie Hertel et al. (2003) gezeigt haben, spielen neben idealistischen Beweggründen auch immer öfter materielle eine Rolle. Die Free Software Foundation schätzt, dass bereits mehr als ein Drittel der Entwicklungsarbeit in allen OSS-Projekten von Unternehmen bzw. deren Mitarbeitern geleistet wird. Die im letzten Abschnitt erwähnten Beispiele machen auch die Hintergründe dieses Engagements deutlich. Nichtsdestotrotz sollte sich im Interesse des Fortbestandes und der Weiterentwicklung der Open-Source-Idee jedes Unternehmen, das freie und Open-Source-Software nutzt, über eine Anerkennung für die erhaltene Software Gedanken machen.

Dass freie und Open-Source-Software generell wieder vom Markt verschwindet, ist also keine realistische Gefahr. Dass ein einzelnes Projekt eingestellt wird, kommt dagegen durchaus häufiger vor. Gerade weniger prominente, kleinere Projekte kämpfen oft ums Überleben. Sie haben zu wenig Zulauf von Entwicklern und können daher ihre selbst gesteckten Ziele nicht einmal erreichen. In einer von uns durchgeführten Umfrage unter verschiedenen Projekten, die in Seifert und Wieland (2003) im Detail vorgestellt wird, wurde der Mangel an Entwicklungskapazitäten gleich von mehreren Befragten genannt. Immer wieder werden Projekte nicht mehr weitergeführt, weil es an Mitarbeitern mangelt. Manchmal entscheidet sich der Projektleiter, der dann ohne Hilfe ist und neben seiner eigenen Arbeitszeit noch die

Serverkosten für das Projekt-Hosting etc. zu tragen hat, auch dafür, das Projekt nur noch als kommerzielle Version weiterzuführen. Je wichtiger einem Nutzer also die OSS-Programme sind, desto mehr sollte er durch eigenes Engagement für einen Fortbestand als Open Source sorgen.

In der öffentlichen Diskussion wird zuweilen auch der „virusartige" Charakter von Open Source als Bedrohung dargestellt. Danach soll durch den Umgang mit OSS auch andere Software des Unternehmens „infiziert" werden, sodass keinerlei Software mehr kommerziell vertrieben werden kann, sobald eine unter einer Open-Source-Lizenz steht (speziell unter der GNU General Public License, GPL). Dies ist natürlich eine völlig falsche und irreführende Behauptung, wie viele Beispiele von IBM über Hewlett Packard bis zu Sun Microsystems zeigen. Der Urheber einer Software behält in jedem Fall das Recht zu entscheiden, unter welcher Lizenz seine Software verbreitet werden soll. Das schließt auch das Recht ein, ein Produkt unter eine Open-Source-Lizenz zu stellen und ein anderes unter eine kommerzielle. Durch die Anwendung einer Open-Source-Lizenz gibt man auch nicht seine Urheberrechte auf – es ist also etwas völlig Anderes als „public domain". Es werden eben nur die Verfügungsrechte des Benutzers anders als bei proprietärer Software geregelt. Gerade aus diesen Gründen beinhaltet auch die GPL die Intention, ihre Lizenz auf von der jeweiligen Software abgeleitete Werke auszudehnen. Wenn ein Programm direkt mit GPL-Software gelinkt wird (im Sinne von Linken mit Objekt-Code), wird dieses als abgeleitetes Werk angesehen und muss es im Quellcode und unter GPL-Lizenz veröffentlicht werden (Free Software Foundation, 1999). Doch in der Praxis besteht nur selten die Notwendigkeit des direkten Linkens mit GPL-Software. Die meisten gebräuchlichen OSS-Systeme, mit denen kommerzielle Software gelinkt ist, stehen unter der so genannten Library GPL (LGPL), die explizit das Linken mit „closed source" erlaubt und die Veröffentlichung des Quellcodes nur bei Änderungen an den Bibliotheken selbst verlangt. Im Übrigen müssen sogar bei GPL-Software die Quellen nur dann offen gelegt werden, wenn die Software selbst veröffentlicht wird. Eine rein firmeninterne Nutzung fällt beispielsweise nicht darunter.

Eine weitere Schwierigkeit beim Umgang mit freier und Open-Source-Software kann aus den gegensätzlichen Einstellungen, Motivationen und Erwartungen der Nutzer beziehungsweise der firmeneigenen Entwickler und den OSS-Entwicklern erwachsen. OSS-Projekte sind üblicherweise bestimmt durch Beiträge zufälliger Programmierer, häufige kleine Releases, Feature-Auswahl nach Bereitschaft und Können der Entwickler, Dezentralisierung sowie „peer review" (Raymond, 1999). Typische Geschäftspraktiken wie Terminpläne, Aufgabenverteilung durch Projektleiter, Festlegung des Funktionsumfangs und des Auslieferdatums durch den Produktmanager sind unter OSS-Entwicklern weitgehend unbekannt (oder werden ignoriert und zuweilen verschmäht). Unternehmen, die sich eine OSS-Community zu Nutze machen und deren Software produktiv einsetzen wollen, tun daher gut daran, diese Spielregeln zu beherzigen und sich mit dieser für sie ungewohnten Kultur auseinander zu setzen (siehe auch Pavlicek 2000). Open-Source-Entwickler akzeptieren es im Allgemeinen nicht, wenn jemand versucht, ihnen seine Entwicklungs- oder Managementprozesse zu oktroyieren. Das bedeutet für die andere Seite, dass

die Firma mehr oder weniger die existierende OSS-Kultur anerkennen und sich um geeignete Schnittstellen zur Community kümmern muss (Näheres bei Seifert und Wieland 2003).

Eines der größten Hemmnisse auf dem Weg zum Einsatz von freier und Open-Source-Software sind gegenwärtig Unsicherheiten bei der Rechtslage. Insbesondere Gewährleistung und Haftung sind offene Punkte, die zwar schon von Juristen mehrfach diskutiert (etwa von Siepmann 1999 sowie von Jaeger und Metzger 2002), bislang aber nicht im Einzelfall gerichtlich entschieden wurden. Man kann sich beispielsweise trefflich darüber streiten, inwieweit etwa der generelle Haftungsausschluss von GPL und anderen Lizenzen mit dem deutschen Recht vereinbar ist – wichtig ist aber zu bedenken, welche rechtlichen Möglichkeiten bei kommerzieller Software gegeben sind. Vor allem entscheidend sind dabei nicht die prinzipiellen rechtlichen Möglichkeiten, sondern ist deren gerichtliche Durchsetzbarkeit.

Für Unternehmer stellt die rechtliche Seite des Einsatz von OSS dennoch eine Unsicherheit dar. Wenn es wirklich um geschäftskritische Anwendungen geht, sollte sich die Firma entweder selbst genügend Know-How zulegen, um den Betrieb in der benötigten Qualität zu gewährleisten, oder damit einen externen Dienstleister betrauen, in dessen Vertrag dann auch Haftungsaspekte aufgenommen werden können. Zudem ist die Lage bei Open Source immer noch deutlich besser, da bei Problemen schnell eigene Verbesserungen vorgenommen oder in Auftrag gegeben werden können. Letztendlich muss man zu bedenken geben, dass auch bei sonstigen Fehlern (und Fehlentscheidungen) im Geschäftsleben das Unternehmen selbst dafür gerade stehen können muss.

Eine generelle Bedrohung von freier und Open-Source-Software ergibt sich aber auch aus möglichen Ansprüchen Dritter an der Urheberschaft. Ein aktuelles Beispiel dafür ist die gerichtliche Auseinandersetzung der Santa Cruz Operation (SCO) mit IBM. Nach Angaben von SCO enthält Linux Codebestandteile, die geistiges Eigentum von SCO sind. Auf Basis dieser Anschuldigung hatte SCO im Juli 2003 Lizenzansprüche vor allem gegen Firmen geltend machen wollen. Jeder Einsatz von Linux solle danach eine Lizenzverletzung sein; SCO hat US-amerikanische Unternehmen bereits angeschrieben und entsprechende Lizenzgebühren eingefordert. Es ist unwahrscheinlich, dass der Streit letztlich zugunsten von SCO ausgehen wird – zu viele Fakten sprechen dagegen. Er zeigt jedoch die Verletzlichkeit der freien und Open-Source-Software. Dass eine bestimmte OSS Rechte Dritter verletzt, die dann auf deren Einhaltung bestehen, ist nie ganz auszuschließen. Für den einzelnen Nutzer handelt sich dabei zwar um ein Risiko, aber doch um ein geringes.

Fazit

Dieser Beitrag kann sicher nur einen ersten Überblick über die Besonderheiten von freier und Open-Source-Software bieten. Die Betrachtung kann daher keinen Anspruch auf Vollständigkeit erheben. Eine Fülle weiteren Materials findet sich in der angegebenen Literatur, dort besonders bei Wheeler (2003).

Die Offenheit, die für Open Source sprichwörtlich ist, stellt für Hersteller wie Anwender eine enorme Chance dar. Hersteller können damit Marketing betreiben

und ein positives Image in der Öffentlichkeit erzeugen, den Markt für weitere Produkte bereiten, Standards setzen oder die Pflege laufender Anwendungen an eine Community übergeben. Gerade der Aufbau und die Motivation einer solchen Community sind jedoch die Aspekte, die über Erfolg oder Misserfolg eines Open-Source-Projekts entscheiden. Auch muss man anders mit Produktzyklen und Zeitplanungen umgehen, als das bei proprietären Systemen üblich ist.

Die Anwender erhalten, wenn sie bereit sind, sich darauf einzulassen, mit dem Quellcode die detaillierteste Dokumentation, die man sich wünschen kann. Abläufe lassen sich so bis auf die unterste Ebene überprüfen, womit sich alle Arten von Fehlern besser aufspüren lassen. Je intensiver der Anwender mit den Quellen arbeitet, desto mehr verschwindet die Grenze zum Herstellerteam. Er erhält also eine neue Rolle, in der er aktiv auf das Produkt Einfluss nehmen kann. Auch wenn damit eine andere Denk- und Handlungsweise verbunden ist, ist Open Source ein höchst interessantes Modell. Der Erfolg von Linux, Apache, MySQL, PHP usw. hat gezeigt, dass es am Durchbruch keine Zweifel mehr geben kann, ja dass dieser gerade schon vor sich geht.

Die gegenwärtigen Ressentiments gehen mehr auf Ängste und Befürchtungen denn auf konkrete Bedrohungen zurück. Jedes Unternehmen muss indessen im Einzelfall entscheiden, ob es für die eine oder andere IT-Aufgabe ein Softwareprodukt kaufen oder eine freie Alternative einsetzen will. Insbesondere der Kosten-Nutzen-Vergleich muss individuell vorgenommen werden und kann nicht durch Modellrechnungen aller Art ersetzt werden. Die freien und Open-Source-Programme sollten allerdings bei jeder Neuanschaffung oder -ausrichtung in Betracht gezogen werden – und das nicht allein aus Kostengründen.

Danksagungen

Der Autor dankt den Kooperationspartnern bei Siemens AG, Siemens Business Services, 4Soft und der Technischen Universität München für die gute und fruchtbare Zusammenarbeit und die anspornenden Diskussionen.

Literatur

Binder, S., C. Lipski (2003): *Kassensturz: Open-Source und proprietäre Software im Vergleich*, Soreon Research

Cybersource (2002): *Linux vs. Windows – Total Cost of Ownership Comparison*, online http://www.cyber.com.au/cyber/about/linux_vs_windows_pricing_comparison.pdf, zuletzt am 20.11.2003 zugegriffen

Free Software Foundation (1999): *Licenses of free software*, online http://www.gnu.org/licenses/, zuletzt am 20.11.2003 zugegriffen

Gillen, A. / Kusnetzky, D / McLarnon, S. (2001): *The Role of Linux in Reducing the Cost of Enterprise Computing*, IDC White Paper, Framingham

Hang, J. / Hohensohn, H. / Mayr, K. / Wieland, T. (2004): *Benefits and Pitfalls of Open Source in Commercial Contexts*, in: Koch, S. (Hrsg.): Free/Open Source Software Development. Hershey, PA

Hars, A., S. Ou (2001): *Working for free? – Motivations for participating in open source projects*, Proceedings 34th HICSS Conference, Maui

Hertel, G., S. Niedner, S. Herrmann (2003): *Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel*, Research Policy, Special Issue

Jaeger, T. / Metzger A. (2002): *Open Source Software – Rechtliche Rahmenbedingungen der Freien Software*, München

Nowak, P. (LW 2003): *The Powerful Economic Underpinnings of OSS*, LinuxWorld 2003, 17.10,
online http://www.linuxworld.com/story/34293.htm, zuletzt am 20.11.2003 zugegriffen

Open Source Initiative (2003): *The Open Source Definition (Version 1.9)*,
online http://www.opensource.org/docs/definition.php, zuletzt am 19.11.2003 zugegriffen

Pavlicek, R. (2000): *Embracing Insanity: Open Source Software Development*, Indianapolis

Rapoza, J. (2002): *eWeek Labs: Open Source Quicker at Fixing Flaws*, eWeek, 30.9.2002
online http://www.eweek.com/article2/0,3959,562226,00.asp, zuletzt am 21.11.2003 zugegriffen

Raymond, Eric S. (1999): *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastopol

Scott, B. (2002): *Why Free Software's Long Run TCO must be lower*,
online http://www.members.optushome.com.au/brendanscott/papers/freesoftwaretco150702.html, zuletzt am 22.11.2003 zugegriffen

Seifert, T. / Wieland, T. (2003): *Prerequisites For Enterprises To Get Involved In Open Source Software Development*, in: 1st Workshop of Open Source Software in Industrial Environments, Proceedings of Net.ObjectDays 2003, Erfurt

Siepmann, J. (1999): *Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software*, LinuxTag 1999,
online http://www.kanzlei-siepmann.de/linux-tag/vortrag.html, zuletzt am 23.11.2003 zugegriffen

Stör, Michael (2003): *Der Migrationsleitfaden*, Bonn

Wheeler, D.A. (2003): *Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!*,
online http://www.dwheeler.com/oss_fs_why.html, zuletzt am 23.11.2003 zugegriffen

Wieland, T. (2001): *Open Source im Unternehmen*, in: A. v. Raison, R. Schönfeldt (Hrsg.): Linux im Unternehmen, Heidelberg