

Open Source

Freie Software als Alternative zu proprietären Entwicklungsmethoden

Patrick Fey

Fakultät für Mathematik, Informatik
und Naturwissenschaften

Universität Hamburg

5fey @ informatik.uni-hamburg.de

ABSTRACT

Open Source Software hat in den letzten Jahren zunehmende Bedeutung erlangt.

Diese Einführung in die Welt des Open Source beleuchtet die Grundlagen, d.h. Philosophie und geschichtliche Entwicklung der freien Softwareentwicklung. Ebenso werden Vorteile des freien Entwicklungsmodells gegenüber klassischen Methoden aufgezeigt.

Es wird jedoch keine vollständige Evaluierung von freien und unfreien Entwicklungsmethoden durchgeführt.

Categories and Subject Descriptors

K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection – *copyrights, licensing, proprietary rights*.

General Terms

Management, Design, Economics, Reliability, Human Factors, Theory, History, Legal Aspects.

Keywords

Open source software, free software, software development.

1. EINLEITUNG

Open Source Software hat in den letzten Jahren einen wahren Boom erlebt. Große Open Source Projekte wie Apache oder Linux und zuletzt Mozilla haben die Entwicklung der Branche wesentlich geprägt und schließlich auch durch ihren Erfolg das Interesse vieler Personen und Organisationen auf sich gezogen, die sich mit Modellen der Softwareentwicklung beschäftigen. Open Source bietet hier einige innovative Ansätze im Vergleich zu proprietären Modellen, die sich auch für die Softwareentwicklung im unternehmerischen Bereich einsetzen lassen.

2. WAS IST OPEN SOURCE?

In den letzten Jahren hat sich die Bezeichnung Open Source oder freie Software für ein bestimmtes Modell bzw. eine bestimmte Philosophie der Softwareentwicklung durchgesetzt. Häufig wird darunter vor allem verstanden, dass der Quelltext öffentlich verfügbar ist und dass die Software kostenlos erhältlich ist.

Um Open Source als Modell der Softwareentwicklung analysieren zu können, ist es jedoch zunächst notwendig, ein tiefergehendes Verständnis für die Terminologie der Open Source Bewegung zu entwickeln. Das ist aber nicht besonders einfach, da sogar in der Open Source Bewegung selbst eine rege Auseinandersetzung darüber herrscht, was der Term 'Open Source' eigentlich bedeutet und ob dieser Term passend ist, um den Entwicklungsprozess und die dahinterstehende Philosophie zu beschreiben.

In den vergangenen Jahren haben sich vor allem zwei Terme und entsprechende Definitionen entwickelt, die in der freien Entwicklergemeinschaft unterschiedlich weit verbreitet sind. Dies sind einerseits der Term "Free Software", der 1985 von Richard M. Stallman und der Free Software Foundation

geprägt wurde [10] und andererseits die vor allem im unternehmerischen Bereich und inzwischen auch außerhalb der freien Entwicklergemeinschaft gebräuchliche Bezeichnung 'Open Source', die Anfang 1998 wesentlich von Eric S. Raymond und Bruce Perens geprägt und in den darauffolgenden Jahren als sogenannte bessere Alternative zu der wesentlich älteren Bezeichnung 'Free Software' propagiert wurde [7].

2.1 Definition von Open Source

Nach der von Bruce Perens und Eric S. Raymond aufgestellten Definition der Open Source Initiative (OSI) [7] ist Open Source Software eine Software, die unter anderem die folgenden Bedingungen erfüllt¹:

2.1.1. Kostenlose Weiterverarbeitung

Open Source Software kann von jedem kostenlos an andere Personen weitergegeben werden.

2.1.2. Verfügbarkeit des Quelltextes

Der Quelltext wird entweder mit dem Programm zusammen verbreitet oder kann von jedem kostenlos aus dem Internet heruntergeladen werden.

2.1.3. Zulässigkeit von abgeleiteten Werken

Andere Entwickler können Modifikationen und abgeleitete Werke von der Software erzeugen und diese zu den gleichen Lizenzbedingungen kostenlos verbreiten.

2.1.4. Keine Diskriminierung von Menschen

Die Software darf in keinem Fall bestimmte Personen oder Personengruppen diskriminieren, z.B. indem es bestimmte Nationen von der Nutzung des Programmes ausschließt.

2.1.5. Keine Diskriminierung von Einsatzfeldern

Die Lizenzbedingungen der Software dürfen in keinem Fall bestimmte Nutzungsfelder für die Software verbieten, z.B. auch nicht den kommerziellen Einsatz oder den Einsatz zur Steuerung von Waffensystemen.

2.1.6. Neutralität gegenüber Technologien

Die Software darf nicht mit einer Beschränkung auf eine bestimmte Anwendungstechnik oder Anwendungsumgebung verbreitet werden. Die Verbreitung von Kopien über normale Datenträger zu verbieten und nur die Verbreitung über das Internet zuzulassen wäre z.B. nicht erlaubt.

Die Open Source Definition nennt als erstes Kriterium die Möglichkeit der kostenlosen Verbreitung. Dies zeigt, dass Raymond und Perens dieses Kriterium besonders wichtig ist.

Obwohl es sich um eine Definition von Open Source Software (deutsch: Software mit offenen Quellen) handelt,

¹ Es werden nur die wichtigsten Eigenschaften von Open Source Software genannt. Für eine genaue Definition siehe [7].

taucht die Anforderung der Verfügbarkeit des Quelltextes erst als zweites Kriterium auf. Die kostenlose Verbreitungsmöglichkeit ist Raymond und Perens also wichtiger, als die Verfügbarkeit des Quelltextes.

Weiterhin wird der Fokus auf die praktischen Aspekte von freier Software bei dieser Definition besonders deutlich. Die ethischen Kriterien (4-6) werden hinter den praktischen Kriterien (1-3) dargestellt.

Die ethischen Kriterien verbieten jegliche Einschränkung der Weiterverbreitungsmöglichkeit. Diese Kriterien unterstreichen damit die Wichtigkeit des ersten Kriteriums der Definition.

2.2 Definition von freier Software

Zum Verständnis der Definition des Terms freie Software von Richard M. Stallman ist es zunächst wichtig, die Bedeutung des Wortes "frei" im Kontext der Definition zu verstehen. Die Free Software Foundation betont darum auch immer wieder, dass "frei" in diesem Kontext nicht als kostenlos zu verstehen ist, sondern vielmehr im Sinne von freiheitlich "*free as in speech, not as in beer*" [10].

Nach der Definition der Free Software Foundation darf diejenige Software als freie Software bezeichnet werden, welche dem Anwender die folgenden vier Freiheiten einräumt:

2.2.1. Unabhängigkeit von Einsatzgebiet (Freiheit 0)

Die Software gibt dem Anwender die Freiheit, sie für jeden möglichen Zweck einzusetzen.

2.2.2. Lernen und Adaptieren (Freiheit 1)

Die Software gibt dem Anwender die Freiheit, ihre Arbeitsweise zu studieren und Änderungen für die eigenen Zwecke vorzunehmen, z.B. eine Funktion hinzuzufügen oder zu ändern.

2.2.3. Weiterverbreitung (Freiheit 2)

Die Software gibt dem Anwender die Freiheit, sie weiterzuverbreiten.

2.2.4. Modifikation und Weiterverbreitung (Freiheit 3)

Die Software gibt dem Anwender die Freiheit, Änderungen an ihr vorzunehmen und diese Verbesserungen zum Wohle der gesamten Gemeinschaft weiterzuverbreiten.

Bei dieser Definition ist die Verfügbarkeit des Quelltextes keine direkte Bedingung für die Bezeichnung einer Software als freie Software. Vielmehr ist es eine Vorbedingung für die Bedingungen 2 und 4, dass der Quelltext verfügbar ist, denn ohne den Quelltext kann weder die Funktionsweise einer Software studiert werden, noch können Modifikationen erstellt werden. Es handelt sich also um eine indirekte Bedingung.

2.3 Unterschiede und Gemeinsamkeiten

Die Unterschiede zwischen der Definition von "Open Source" und der Definition von "Free Software" sind in der Praxis von wenig Bedeutung. Beide Definitionen enthalten im Prinzip die selben Kriterien, wenngleich sie unterschiedlich formuliert wurden. Viele einschlägige Software-Lizenzen sind von beiden Definitionen abgedeckt. Die meiste Software, die in der freien Entwicklergemeinschaft entwickelt wird, ist daher sowohl "Open Source", als auch "Free Software". Ob ein Entwickler Open Source Software oder freie Software entwickelt, spielt für ihn daher kaum eine Rolle. In beiden Fällen hat er die gleichen Kriterien zu erfüllen [5: 79].

Die Open Source Initiative legt den Fokus auf die praktischen Aspekte. Der ethische Teil der Definition wird erst später genannt. Dies ist vor allem darin begründet, dass die Open Source Initiative mit ihrer Terminologie vor allem

Unternehmen von den praktischen Vorteilen des Open Source Softwareentwicklungsmodells überzeugen will. Darum wurde der Term "Open Source" von der Open Source Initiative auch als Handelsmarke eingetragen.

Im Gegensatz dazu spricht die Free Software Foundation von Freiheiten und legt den Fokus vor allem auf die politischen Aspekte und Vorteile von Open Source. Die Free Software Foundation möchte mit ihrer Definition herausheben, in welchem Maße Software heute den Alltag vieler Personen bestimmt und warum es daher wichtig ist, dass die Software, die die Gemeinschaft kontrolliert, auch unter der Kontrolle der Gemeinschaft steht [5: 79]. Praktische Überlegungen, wie die Verfügbarkeit des Quelltextes, sind nur indirekte Bedingungen und nicht Teil der eigentlichen Definition.

Es ist daher heute vor allem eine politische Entscheidung, ob ein Entwickler von seiner in der freien Entwickler-Gemeinschaft geschriebenen Software als freie Software oder als Open Source Software spricht.

2.4 Geschichtliche Entwicklung

Eric S. Raymond behauptet, dass sich das Prinzip von Open Source Software aus den Wurzeln des Internet in den 1970er Jahren entwickelt hat [9]. Es ist zwar richtig, dass das Internet für die Entwicklung von Open Source eine wichtige Rolle gespielt hat, da erst das Internet eine entscheidende Stärke von Open Source entfalten konnte: Die extrem schnelle Entwicklung von Software unter Einbeziehung möglichst vieler qualifizierter und hoch motivierter Personen zur gegenseitigen Kontrolle und zur Qualitätssicherung [2: 181]. Auch wurde das Internet nicht zuletzt durch Open Source selbst erst zu dem, was es heute ist: Die meisten Standards, die für den technischen Betrieb des Internet genutzt werden, sind als offene Standards entwickelt worden. Auch heute wird ein Großteil der Internetinfrastruktur mit Open Source Software betrieben (z.B. Apache Webserver, Bind Nameserver, Sendmail Mailserver, u.a.) [6: 34]. Das Internet ist damit einerseits Teil des zunehmenden Erfolges von Open Source Anwendungen, aber andererseits gleichzeitig auch Nutznießer von Open Source.

Nach Glass reichen die Ursprünge der Open Source Bewegung aber bis in die Pionierzeit des Computerzeitalters in den 1950er Jahren, d.h. wesentlich weiter, zurück [4: 26]. Damals wurde Software meist kostenlos verbreitet. Software wurde noch nicht als wertvoll angesehen. Darum wurde Software in der Regel in Form von offenem Quellcode verbreitet. Insbesondere im wissenschaftlichen Bereich wurde Software - in der Regel über spezielle Bibliotheken oder Software-Banken - gesammelt, verbreitet und ausgetauscht. Bis in die 1960er/1970er Jahre war Software demnach standardmäßig in einer Form in der Branche verfügbar, die dem heutigen Begriff des Open Source bzw. dem Begriff der freien Software sehr nahe kommt [3: 59].

1969 kommt es jedoch zu einem Ereignis, welches laut Glass [4: 26] als einschneidend für die weitere Entwicklung auf dem Software-Markt, insbesondere im Rahmen der Entwicklung von proprietärer und freier Software gesehen werden kann. Das U.S. Department of Justice reicht eine Klage gegen den damaligen Marktführer IBM ein, in der unter anderem die Frage geklärt werden sollte, ob die Bündelung von Software und Hardware zu gemeinsamen Paketen, wie sie bisher in der Branche üblich und neben IBM auch von anderen Unternehmen praktiziert worden war, gegen den Artikel 2 des Sherman Act verstöße, nachdem eine Monopolisierung eines freien Marktes von einem Unternehmen nicht angestrebt werden darf. Auf Grund des weiteren Verlaufs des Prozesses, der bis Mitte der 1970er Jahre andauerte, war IBM schließlich gezwungen, Hardware und Software getrennt anzubieten. Obwohl IBM nachgesagt wurde, die Software weit unter Wert

anzubieten, um die Etablierung eines freien Marktes zu verhindern, führte dies schließlich dazu, dass sich in den nächsten Jahren zunächst kleinere Softwareunternehmen am Markt durchsetzten und am Ende auch größere Unternehmen wie Microsoft entstanden. Langfristig wandelte der Prozess gegen IBM damit die ursprüngliche allgemeine Ansicht, nach der Software keinen Wert hatte und daher in Form von offenen Quellen verbreitet werden konnte, zum heutigen Verständnis des Begriffs Software, nach dem Software eine Dienstleistung ist, welche Geld kostet, und damit zwingend mit geschlossenen Quellen verbreitet werden muss, um den Quelltext der Software als das sogenannte intellektuelle Eigentum (*intellectual property, IP*), welches Profit und somit einen *Return On Investment (ROI)* erst ermöglicht, vor Missbrauch durch Dritte zu schützen [2: 178].

Nicht zuletzt für Firmen wie Microsoft, die quasi ein reines Produktportfolio aus Software haben, ist der Schutz ihres intellektuellen Eigentums ein wichtiger Grundsatz.

Dieses neue Verständnis und der sich neu entwickelnde Markt für Software bewirkten aber auch, dass sich in den folgenden Jahren immer wieder Initiativen bildeten, die eine Rückkehr zu alten Verarbeitungsformen für Software und einem offenen Austausch propagierten. Diese Initiativen bildeten sich häufig im akademischen Bereich, wie z.B. die Berkeley System Distribution an der Universität Berkeley in den USA [3: 59].

1983 hatte Richard M. Stallman mit der Entwicklung eines freien Unix-Derivats begonnen. Sein GNU (Gnu's not Unix) getauftes Projekt brachte eine Reihe von freien nützlichen Tools für Unix hervor. 1985 gründete er die Free Software Foundation und sorgte so dafür, dass sich in den folgenden Jahren der Ausdruck "Freie Software" für sein Projekt und ähnliche Softwareprojekte in der Entwicklergemeinschaft durchsetzte. Neben der Definition dieses Ausdrucks ging aus der Free Software Foundation auch die noch heute am weitesten verbreitete Lizenz für freie Software, die *GPL General Public License*, hervor.

Das vor allem in angelsächsischen Sprachraum bestehende Problem dieses Terms versuchte Stallman durch die genauere Definition des Ausdrucks zu vermeiden, in der er festlegte, dass "frei" in diesem Zusammenhang als freiheitlich, nicht jedoch als kostenlos zu verstehen sei [10].

In 1998 geriet die Firma Netscape, die den bis dahin sehr beliebten Browser 'Netscape Communicator' anbot, in eine Krise, nachdem Microsoft sein Konkurrenzprodukt 'Internet Explorer' zusammen mit seinem Betriebssystem Windows anbot und so einen Großteil der Marktanteile für Internet Explorer sichern konnte. Netscape entschloss sich daher, den gesamten Quelltext seines Browsers unter einer Open Source Lizenz zu veröffentlichen und hoffte, sich auf diese Weise neue Marktanteile sichern zu können.

Dieser mutige und damals noch ungewöhnliche Schritt von Netscape erregte sowohl in der freien Entwicklergemeinschaft als auch in der Software-Branche eine Menge Aufsehen und veranlasste auch andere Firmen dazu, ihre Einstellung gegenüber freier Software in den nächsten Jahren neu zu überdenken [3: 60].

Bruce Perens, Eric S. Raymond und andere beschäftigten sich als Folge der Entscheidung von Netscape mit der Attraktivität von freien Softwareentwicklungsmodellen für Unternehmen und kamen schließlich überein, dass die Definition des Terms freie Software ideologisch zu überladen und auch auf Grund der Doppelbedeutung im angelsächsischen Sprachraum sehr missverständlich war. Beides war aus ihrer Sicht nicht gerade förderlich für eine unternehmerische Nutzung von freier Software, so dass sie schließlich noch im selben Jahr die Open Source Initiative (OSI) gründeten und eine Definition für einen neuen Ausdruck "Open Source" (deutsch: offene Quellen) niederschrieben [7]. Durch den Fokus auf die praktischen Vorteile von Open Source als Entwicklungsmodell sollte eine

stärkere Verbreitung im unternehmerischen Bereich und somit eine Verbreitung von Open Source Software im Hauptanwender-Markt (engl. *mainstream*) ermöglicht werden [8, 3: 59].

Diese Umbenennung von freier Software hat seitdem immer wieder zu Konflikten in der Entwicklergemeinschaft geführt, auch wenn dadurch die praktische Entwicklung nicht behindert wurde. Richard M. Stallman sieht dabei die Open Source Bewegung nicht als 'Feind' an. Der gemeinsame 'Feind' der Open Source und Free Software Bewegungen sei vielmehr proprietäre Software. Die freie Software Bewegung erkenne die Beiträge der Open Source Bewegung zur freien Software-Gemeinschaft an. Insbesondere wolle man jedoch, dass die freie Software-Gemeinschaft auch mit den Werten und der Philosophie der freien Software Bewegung assoziiert werde, nicht mit denen der Open Source Bewegung. Schließlich habe Stallman selbst die freie Entwicklergemeinschaft gegründet und aufgebaut [11].

3. OPEN SOURCE ALS MODELL FÜR DIE SOFTWAREENTWICKLUNG

3.1. Klassische Softwareentwicklung

Um ein hinreichendes Verständnis für die Vorteile von Open Source als Modell für die Softwareentwicklung zu erhalten, müssen zunächst einige Aspekte der klassischen, proprietären Softwareentwicklung genauer betrachtet werden.

Seit Beginn der 1970er Jahre [1] wurde im wissenschaftlichen Umfeld hauptsächlich der Term "Software-Krise" gebraucht, um einige wesentliche Probleme der proprietären Softwareentwicklung zu beschreiben. Insbesondere sollten hier drei große Problemkomplexe zusammengefasst werden:

1. Die Entwicklung proprietärer Software dauert in der Regel zu lange.
2. Proprietäre Software ist in der Regel zu teuer.
3. Proprietäre Software erfüllt in der Regel die an sie gestellten Anforderungen nicht zufriedenstellend.

Diese Probleme konnten trotz zahlreicher vorgeschlagener Lösungsansätze bisher nicht zufriedenstellend gelöst werden [3: 58]. Vor allem in größeren Unternehmen, die oft mehrere Abteilungen mit insgesamt tausenden von Entwicklern beschäftigen, kommt es zu charakteristischen Erscheinungen, die bei genauerer Betrachtung mit als Auslöser und Förderer dieser Problemkomplexe gesehen werden können:

So gibt es beispielsweise häufig keine sog. *Coding Standards*, die konsistent im gesamten Unternehmen eingesetzt werden. Jede Abteilung definiert eigene Regeln und setzt eigene Werkzeuge zur Softwareentwicklung ein. Insbesondere in Verbindung mit den für Software-Unternehmen typischen plötzlichen Umstrukturierungen von Entwickler-Teams und verschiedenen Abteilungen kann das einerseits zu erheblichen Qualitätseinbußen, aber auch zur Verlängerung der Entwicklungszeit und damit der Verteuerung des Produktes führen [2: 177].

Durch mangelnde Kommunikation zwischen den Abteilungen kommt es auch häufig zu sog. Parallelentwicklungen. So arbeitet eine Abteilung z.B. an einer Problemstellung, die in dieser oder ähnlicher Form zeitgleich von einer anderen Abteilung bearbeitet wird oder die sogar von einer anderen Abteilung bereits erfolgreich gelöst wurde. Diese fehlenden Synergieeffekte zwischen den verschiedenen Abteilungen verlangsamten die Entwicklung erheblich und tragen ebenso zur Erhöhung des Kostenfaktors bei [2: 177].

Ein weiteres Problem in der klassischen Softwareentwicklung ist auch der eingeschränkte Kommentar- und Überprüfungs-

prozess, insbesondere in kleineren Entwicklergruppen. Vor allem in den frühen Entwicklungsphasen, die auf Grund der Festlegung der Software-Architektur und der Entscheidung über die Funktionen und Eigenschaften des Programmes besonders wichtig für eine weitere positive Entwicklung der Anwendung sind, beschränken sich die Prozesse des gegenseitigen Austausches auf den kleinen Kreis des eigenen Entwickler- und Marketing-Teams [2: 178].

Die hierarchische Organisationsstruktur in Unternehmen führt häufig auch zu Behinderungen des natürlichen Prozesses der Softwareentwicklung. So können Entscheidungsträger, die mit dem Entwicklungszustand der jeweiligen Software oder sogar mit den eigentlichen Grundprinzipien der Softwareentwicklung nicht oder nur wenig vertraut sind, dennoch erheblich in den Entwicklungsprozess eingreifen, um beispielsweise ein vom Marketing versprochenes Lieferdatum einhalten zu können, ohne dass zuvor die gewünschte Qualitätsstufe einer Funktion oder der gesamten Anwendung erreicht wurde [2: 179].

3.2. Freie Softwareentwicklung

Freie Softwareentwicklung zeichnet sich in den meisten Fällen durch vier wesentliche Punkte aus:

1. Schneller Entwicklungsprozess.
2. Hohe Qualität des Endproduktes.
3. Hohe Wiederverwendbarkeit des Quelltextes.
4. Niedrige Entwicklungskosten.

Insbesondere diese vier Punkte machen die Methoden der freien Softwareentwicklung für Firmen attraktiv, da diese Methoden eine mögliche Lösung für die Probleme klassischer Softwareentwicklung darstellen können [3: 63].

Bereits die Herangehensweise der freien Entwicklergemeinschaft an ein Projekt unterscheidet sich grundlegend von der Herangehensweise klassischer Entwicklungsteams.

Im kommerziellen Umfeld wird Software in der Regel als ein Produkt gesehen, welches einen möglichst hohen Profit und somit einen schnellen und hohen *Return on Investment (ROI)* ermöglicht. Im Open Source Bereich kommen dagegen vor allem Anwendungen zur Entwicklung, die ein spezielles Problem eines Entwicklers lösen. Der *ROI* wird dabei durch die Problemlösung repräsentiert. Dafür, dass der Entwickler sein Projekt anderen gleichgesinnten Entwicklern zur Verfügung stellt, erhält er u.U. neue auch ihm nützliche Funktionen für seine Anwendung. Die geschichtliche Entwicklung vieler Open Source Projekte zeigt, dass sie ursprünglich zur Lösung eines speziellen Problems entwickelt wurden und erst mit der Zeit die Bedeutung in der Branche erhielten, die sie heute haben (z.B. Sendmail) [6: 35].

Aus dieser Tatsache heraus hat sich langfristig eine flache, d.h. nicht hierarchische Organisationsstruktur entwickelt, die eine besonders schnelle, zielorientierte Softwareentwicklung fördert. Dabei übernehmen eine oder wenige besonders qualifizierte Personen die Betreuung des Projektes. Sie sind die sog. *Maintainer* der Software und bestimmen, welcher Quelltext gut genug ist, um in den Quelltext der Anwendung übernommen zu werden. Um die Maintainer herum versammeln sich eine große Anzahl von gleichgestellten Entwicklern, die dezentralisiert und unabhängig von einander in Zusammenarbeit mit jeweils gleichgesinnten Entwicklern an den Funktionen arbeiten, die sie selbst benötigen [2: 181, 3: 65].

Um einen möglichst großen Pool von Entwicklern zu erreichen, stehen Quelltext und die benötigten Werkzeuge in der Regel im Internet frei zur Verfügung. Durch die flache Organisationsstruktur, die dem Prinzip eines Ameisenhaufens (eine Königin, tausende Arbeiter mit jew. kleinen Aufgaben)

ähnelt, können Entwickler mit unterschiedlichem Wissensstand und aus unterschiedlichen Zeitzonen in das Projekt integriert werden [3: 64, 6: 36]. Durch die große Anzahl an potentiellen Entwicklern wird der Entwicklungsprozess beschleunigt.

Auf Grund dieser dezentralen Entwicklungs- und Organisationsstruktur kann eine Kommunikation der Entwickler untereinander oft nur per Email erfolgen.

Da sich die Entwickler meist nicht persönlich, sondern nur über den Kontakt per Email oder Diskussionsforen, kennen, ist der Ruf des einzelnen Entwicklers in der Gemeinschaft entscheidend von der Qualität der von ihm gemachten Beiträge in Form von Quelltext o.ä. abhängig [3: 65].

Neben der eigentlichen Entwicklungsarbeit findet in der Regel ein ständiger, offener Diskussions- und Kontrollprozess statt. Die Entwickler schauen sich den Quelltext ständig an und diskutieren über neue Beiträge von anderen Entwicklern oder über neue Funktionen für die Software. Dadurch fallen Qualitätsmängel oder Sicherheitslöcher meist schnell auf und werden zügig behoben [2: 183].

Die Abhängigkeit des Rufs in der Gemeinschaft von der Qualität der eigenen Beiträge einerseits sowie der offene Diskussionsprozess und die andauernde gegenseitige Kontrolle andererseits sind vergleichbar mit üblichen wissenschaftlichen Arbeitsmethoden und tragen entscheidend zur Qualität der Software bei.

Entscheidend für die hohe Qualität von freier Software ist außerdem die funktions-fokussierte Entwicklungsmethode. Die Entwicklung der Software orientiert sich in der Regel nicht an zeitlichen Abläufen, sondern es sollen bestimmte Probleme der Entwickler gelöst werden, d.h. die Entwicklung orientiert sich an bestimmten Funktionen, die der Entwickler implementiert haben will. Die entwickelte Software ist erst dann erfolgreich implementiert, wenn das Problem des jew. Entwicklers hinreichend gelöst wurde.

Je nachdem, welche neuen Anforderungen sich in der Nutzergemeinschaft ergeben, wird sich die Software unterschiedlich entwickeln. Dies entspricht einem natürlichen, evolutionären Entwicklungsprozess, der einen hohen Nutzen der Software für die jeweiligen Anwender sicherstellt.

Dadurch, dass der Quelltext der verschiedenen Open Source Projekte öffentlich verfügbar ist, ist gleichermaßen eine hohe Wiederverwendbarkeit des Quelltextes gewährleistet. Ein neues Open Source Projekt kann zur Lösung bestimmter Problemstellungen auf einen großen Pool an frei verfügbaren Quelltext zurückgreifen, den es für seine Zwecke übernehmen oder modifizieren kann, ohne dabei großen Entwicklungs- und Planungsaufwand zu haben. Die Möglichkeit der Verwendung von bereits bestehendem und getesteten Quelltext verbessert auch wiederum die Qualität der Software [2: 179].

Der schnelle Entwicklungsprozess, die hohe Qualität und nicht zuletzt die hohe Wiederverwendbarkeitsrate des Quelltextes tragen schließlich alle entscheidend dazu bei, die theoretischen Entwicklungskosten vergleichsweise gering zu halten.

3.3. Freie Software in Unternehmen

In einer kommerziellen Umgebung ist der Schutz des eigenen intellektuellen Eigentums als das "Kapital" des Unternehmens sicherlich ein wichtiger Aspekt. Auf Grund der Überlegungen, die daraus entstehen, können freie Entwicklungsmodelle deshalb auch nur bedingt eingesetzt werden.

Dennoch sind die Möglichkeiten, die Open Source bietet, nämlich der hohe Qualitätsstandard der Software und der hohe Innovationsfaktor während der Entwicklungsphase, für Unternehmen heute nicht mehr zu umgehen.

So besteht zum Beispiel die Möglichkeit, die Vorteile von Open Source nur innerhalb des eigenen Unternehmens zu nutzen, indem der gesamte Quelltext der Produkte eines

Unternehmens allen Entwicklern des Unternehmens zur Verfügung steht und darauf die Prinzipien der freien Softwareentwicklung angewendet werden [2: 178].

Eine andere Möglichkeit ist, nur einige Teile des Produktportfolios als Open Source zur Verfügung zu stellen und so in diesem Bereich die Vorteile von Open Source für die eigene Produktentwicklung zu nutzen.

Im Open Source Bereich selbst haben sich Unternehmen etabliert, die ein durchgehend neues Geschäftsmodell entwickelt haben. Diese Unternehmen verstehen sich selbst als Dienstleister, die dem Kunden ergänzende Dienstleistungen aufbauend auf die eigentlich kostenlose Software verkaufen. Es entsteht ein Wettbewerb auf Basis der Güte der Dienstleistungen, die ein Unternehmen den Kunden anbietet. Nach Boehm und Flaatten (1976, 1989) wird ein Hauptteil der Software-Entwicklungskosten während der Kundenbetreuung verursacht und nicht bereits während der ursprünglichen Entwicklungsphase. Das o.a. neue Geschäftsmodell entspricht daher eher den ökonomischen Erfordernissen.

Ein großer Vorteil dieses neuen Geschäftsmodells ist auch, dass es den jeweiligen Unternehmen ermöglicht, Risiken und langfristige Entwicklungskosten, aber auch eigene positive Entwicklungen mit anderen interessierten Unternehmen zu teilen. Letztendlich profitieren alle Unternehmen nicht in erster Linie durch den Erfolg der eigenen Distribution, sondern durch den Erfolg der gesamten Plattform, auf der die Distributionen der einzelnen Unternehmen aufbauen [3: 64].

5. ABSCHLUSS

In der freien Entwicklergemeinschaft haben sich über die Jahre verschiedene Definitionsmöglichkeiten für Open Source Software herausgebildet. In der Praxis macht die Art der Definition für den Entwickler aber keinen Unterschied.

Die Ursprünge von Open Source liegen in den 1950er Jahren. Für Firmen ist Open Source aber erst 1998 richtig interessant geworden.

Open Source löst klassische Probleme der proprietären Softwareentwicklung, wie lange Entwicklungsdauer, hohe Kosten und Nichterfüllung von Anforderungen der Nutzer. Im unternehmerischen Bereich ist es aber wichtig, sein eigenes intellektuelles Eigentum zu schützen, darum ist die Anwendbarkeit eingeschränkt.

In Zeiten einer steigenden Notwendigkeit von globaler Zusammenarbeit (Globalisierung) ist Open Source vor allem für große Unternehmen ein interessantes Vorbild und ein innovativer Ansatz, dessen (Teil-)Nutzung zahlreiche Vorteile für das eigene Unternehmen mit sich bringen kann.

4. REFERENZEN

- [1] DIJKSTRA, Edsger W. The Humble Programmer. *Communications of the ACM*, 15, 10, 1972, 859-866.
- [2] DINKELACKER, Jamie; GARG, Pankaj K.; MILLER, Rob; NELSON, Dean. Progressive Open Source. *ICSE*, Orlando, Florida, USA, 2002, 177-184.
- [3] FELLER, Joseph, FITZGERALD, Brian. A framework analysis of the open source software development paradigm. *CIS*, 2000, 58-69.
- [4] GLASS, Robert L. A Look at the Economics of Open Source - Is open source the future of the software field or a passing fad? *Communications of the ACM*, 47,2, Feb. 2004, 25-27.
- [5] JOHNSON-EILOLA, Johndan. Open Source Basics - Definitions, Models, and Questions. *SIGDOC*, Toronto, Ontario, Canada, October 20-23, 2002, 79-83
- [6] O'REILLY, Tim. Lessons from Open Source Software Development. *Communications of the ACM*, 42, 4, Apr. 1999, 33-37.
- [7] PERENS, Bruce. The Open Source Definition. <http://opensource.org/docs/definition.php>, am 15. November 2005.
- [8] RAYMOND, Eric S. Goodbye, "free software"; hello, "open source". <http://www.catb.org/~esr/open-source.html>, am 15. November 2005.
- [9] RAYMOND, Eric S. Why you should care. <http://www.catb.org/~esr/writings/cathedral-bazaar/introduction/>, am 15. November 2005
- [10] STALLMAN, Richard M. The Free Software Definition. <http://www.fsf.org/licensing/essays/free-sw.html>, am 15. November 2005.
- [11] STALLMAN, Richard M. Why "Free Software" is better than "Open Source". <http://www.fsf.org/licensing/essays/free-software-for-freedom.html>, am 15. November 2005.