Kapitel 3

Technik

Einleitung

YACINE GASMI

Die große Wirtschaft hat die Open-Source-Bewegung lange Zeit für uninteressant gehalten, weil sie schlicht keine konkurrenzfähigen Produkte von einer Gemeinschaft erwartete, in der Software als Hobby programmierfreudiger Individuen und ohne unternehmerisches Dachwerk entstand. Open Source galt mehr als Spielwiese für Idealisten denn als Umfeld für professionelle Softwareentwicklung.

Durch den sichtbaren Erfolg zahlreicher Projekte, allen voran Linux, hat in den letzten Jahren jedoch ein Umdenken stattgefunden. Branchenriesen wie IBM oder SAP setzen zunehmend auf diesen alternativen Weg der Softwareentwicklung. Auf IBMs Inititiative hin entstand das Eclipse-Projekt, in dem namhafte Softwareunternehmen an der Herstellung einer Open-Source-Programmierumgebung arbeiten.[1] Und SAP, dessen Datenbank SAP DB seit Oktober 2000 unter einer Open-Source-Lizenz vertrieben wird, ist im Mai 2003 eine Partnerschaft mit dem schwedischen Unternehmen MySQL AB eingegangen, um gemeinsam eine Open-Source-Datenbank zu entwickeln.[2] Diesen Trend hin zu einer Öffnung gegenüber dem Open-Source-Gedanken verdeutlicht auch das Engagement diverser IT-Branchenführer in den *Open Source Development Labs (OSDL)*. Die OSDL wurden 2000 durch ein Konsortium der Unternehmen IBM, HP, CA, Intel und NEC gegründet. Zu den Mitgliedern zählen inzwischen über 30 namhafte Firmen aus dem IT-Sektor.[3] Durch die Unterstützung geeigneter Projekte mittels Bereitstellung von Entwicklungs- und Testumgebungen sollen Enterprise-Lösungen für Linux entwickelt und somit dessen Einsatz in Unternehmen forciert werden.

Die Beweggründe für das Engagement der genannten Firmen mögen vorrangig ökonomischer Art sein und sollen an dieser Stelle auch nicht diskutiert werden. Die genannten Beispiele zeigen aber, dass Freie Software[4] inzwischen ernst genommen wird, da man erkannt hat, dass diese Art und Weise der Softwareentwicklung durchaus qualitativ hochwertige Produkte hervorbringen kann. Wofür aber steht „Open

---

1 Die Homepage des Eclipse-Projekts findet sich unter http://www.eclipse.org.

2 Für die Pressemitteilung von SAP siehe Literaturverzeichnis: SAP 2003.

3 Die komplette Liste der beteiligten Unternehmen ist unter http://www.osdl.org/about_osdl/members/ zu finden.

4 „Freie Software“ und „Open-Source-Software“ werden hier stets im Sinne von quelloffener Software verwendet. Die genaue begriffliche Unterscheidung wird im Einführungskapitel ausführlich behandelt.

Source“ eigentlich genau? Der Begriff an sich benennt (auf Software bezogen) streng genommen lediglich eine Lizenzierungsform und ist in diesem Sinne keine Entwicklungsmethode. Die Offenlegung und die mit der spezifischen Lizenzierung verbundene freie Verfügbarkeit und Wiederverwendbarkeit des Quellcodes führen allerdings zu bestimmten Charakteristika bei der Entwicklung Freier Software. Beispielsweise ist das Mitwirken beliebig vieler Programmierer an einem Projekt nur dadurch möglich, dass jeder *im Prinzip* uneingeschränkten Zugang zum Programmquellcode hat. Betrachten wir deshalb an dieser Stelle der Einfachheit halber Open Source als ein bestimmtes Entwicklungsmodell und die proprietäre Variante der Softwareherstellung als das entsprechende Gegenmodell.[5] Dann liegt die Frage nahe, welches von beiden denn das bessere Modell ist. Oder wenn man etwas zugespitzt die Zukunft der Softwareentwicklung ins Spiel bringt: Wird sich ein Modell gegen das andere durchsetzen? Sofort ertönen die Prophezeiungen der ideologischen Hardliner auf beiden Seiten, die in der Open-Source-Bewegung entweder eine bald wieder verpuffende Modeerscheinung sehen oder entsprechend im anderen Lager das Ende der proprietären Softwareentwicklung verkünden. Die Wahrheit wird wie so oft irgendwo dazwischen zu finden sein.

Mit diesem Jahrbuch soll versucht werden, den Leser in seiner Urteilsbildung bezüglich der Möglichkeiten und Grenzen der Open-Source-Bewegung zu unterstützen. Die Worte „Möglichkeiten“ und „Grenzen“ sind ganz bewusst gewählt, denn letzten Endes spielen zu viele Faktoren eine Rolle im Entwicklungsprozess quelloffener Software, als dass ein Pauschalurteil über die Qualität von Open-Source-Software als solcher gefällt werden könnte.

Man tut sich ohnehin schwer, die Qualität von Software zu messen. Auf Grund ihrer Komplexität ist Software nicht greifbar und daher nur schwer zu beurteilen. Als Hilfestellung und Grundlage für eine Bewertung hat man bestimmte Anforderungen formuliert: Die wesentlichen sind Zuverlässigkeit, Wartbarkeit, Effizienz und Benutzerfreundlichkeit (Sommerville 2001, S. 28). Je nach Erfordernissen an die Anwendung fällt die Gewichtung unterschiedlich aus. Ein Banksystem beispielsweise sollte vor allem zuverlässig und insbesondere sicher sein, während bei einem Computerspiel die Benutzerfreundlichkeit einen hohen Stellenwert besitzt. Die genannten Qualitätskriterien bilden zwar eine Art Gerüst für die qualitative Analyse von Software, die letztendliche Bewertung eines Programmes wird dadurch aber nicht wesentlich einfacher. Einen Weg aus dem Dilemma bietet die Analyse und Bewertung des Entstehungsprozesses, d.h., man untersucht, wie die Software hergestellt wurde, und zieht dann Schlüsse auf die Qualität des Endproduktes. Dieser Ansatz hat zur Spezifizierung verschiedener Qualitätsstandards geführt – die bekann-

[5] Dies können wir ohne Gewissensbisse tun, da selbst Eric S. Raymond in seinem berühmten Artikel: „The Cathedral and the Bazaar“ so verfährt. Auch auf die unterschiedlichen herkömmlichen Vorgehensmodelle (z. B. Wasserfall-Modell) bei der Entwicklung proprietärer Software soll hier nicht näher eingegangen werden, vergleiche dazu (Sommerville 2001).

testen dürften die „ISO 9000"-Reihe[6] und das *Capability Maturity Model (CMM)*[7] sein – welche konkrete Anforderungen an den Entwicklungsprozess stellen.

In diesem Kapitel wollen wir analog verfahren und die Betrachtung der Freien Software unter technischen Gesichtspunkten mit einem Einblick in den Entstehungsprozess beginnen.

Den Anfang macht *Matthias Ettrich*. Als Gründer des *K Desktop Environment (KDE)* hat er den Aufstieg eines inzwischen internationalen Projektes mit über tausend freiwilligen Mitarbeitern hautnah von Anfang an miterlebt. Im ersten Beitrag dieses Kapitels schildert er den Entstehungsprozess quelloffener Software unter dem Aspekt der „Koordination und Kommunikation in Open-Source-Projekten." Die anfangs erwähnte Skepsis gegenüber quelloffener Software ist u.a. auf der Vorstellung begründet, dass durch das (scheinbar chaotische) Zusammenwirken sehr vieler Mitarbeiter, die zudem meist Hobbyprogrammierer sind, kein „professionelles" Produkt entstehen kann. Der Gedanke, dass sich die Vergrößerung des Teams in der Regel nicht förderlich auf den Entwicklungsprozess auswirkt, wurde bekannt als Brook's Law: „Adding manpower to a late software project makes it later." (Brooks, Jr. 1975; zit. n. González-Barahona 2003, S. 52). Open-Source-Projekte wie Linux schöpfen ihre Dynamik aber gerade aus der großen Anzahl an Beteiligten und ihren unteschiedlichen Kompetenzen. Um dieses Potenzial produktiv umsetzen zu können, müssen die vielen separaten Anstrengungen einzelner Personen koordiniert werden. Der Rolle der Projektleitung kommt dabei eine für den Erfolg des Projektes entscheidende Bedeutung zu, und so setzt sich Matthias Ettrich in seinem Artikel intensiv mit den Anforderungen an diese auseinander.

Als vielleicht wichtigste, da für das Projekt lebensnotwendige Aufgabe kann man die Motivierung der Entwicklergemeinschaft betrachten. Vergleicht man zwei der erfolgreichsten Open-Source-Projekte miteinander, Linux und Apache, dann stellt man fest, dass durchaus unterschiedliche Führungsstile zum Ziel führen können. Während Linus Torvalds, Gründer und Leiter des Linux-Projektes, den Dingen eher freien Lauf lässt, wird das Apache-Projekt von einem unternehmensartig organisierten Führungsgremium dominiert. Hier müssen sich Entwickler über lange Zeit bewähren, um in den elitären Kreis der Hauptentwickler aufgenommen zu werden, welcher für den größten Teil des geschriebenen Codes verantwortlich zeichnet. Der Großteil der Community ist grundsätzlich selten direkt an der Erarbeitung neuen Codes beteiligt. Ihr bedeutsamer Beitrag liegt vielmehr im Testen der Software sowie in der Fehlerfindung und -behebung. Ettrich nennt in seinem Text unterschiedliche Möglichkeiten der Mitarbeit und zeigt auf, welche Bedeutung das Internet für die Kommunikation zwischen den Projektteilnehmern erlangt hat. Abschließend kommt er noch auf ein heikles Thema der Open-Source-Bewegung zu sprechen, das Spalten eines Projektes:

---

6 Internationaler Industriestandard für die Entwicklung eines Qualitätsmanagementsystems in einer Organisation. Mehr zu ISO 9000 unter http://www.iso.ch/iso/en/iso9000-14000/index.html.

7 Von Carnegie Mellon's *Software Engineering Institute (SEI)* zur Verbesserung der Prozessqualität entwickelt. Bewertung des „Reifegrads" (engl.: *maturity*) einer Organisation nach 5-stufigem Modell. Näheres unter: http://www.sei.cmu.edu/cmm/cmm.html.

Da die Mitarbeit an einem Projekt freiwillig geschieht, kann ein unzufriedener Entwickler jederzeit abspringen. Die freie Verfügbarkeit des Quellcodes ermöglicht es ihm zudem, auf eigene Faust weiterzumachen und ein eigenes Projekt von dem ursprünglichen abzuzweigen. Dieses Vorgehen wird *forking* genannt und ist nicht ganz unproblematisch für die Open-Source-Gemeinschaft. Durch *forking* entstandene verwandte Projekte können sich im besten Fall zwar in völlig verschiedene Richtungen entwickeln, sie können aber auch zu Konkurrenten werden. Dem weit verbreiteten Argument, *forking* führe lediglich zu einer Art darwinistischem Selektionsprozess, bei dem sich die interessanten Projekte durchsetzen, stehen kritische Stimmen wie Nikolai Bezroukov (1999) gegenüber, die die Konkurrenzsituation für existenzbedrohend (für die gesamte Bewegung) halten – auf Grund begrenzter „Human Ressources", um die gebuhlt wird. Matthias Ettrich zeigt, welche Arten des *forking* existieren und welche Folgen aus ihnen resultieren.

Während Ettrich sehr praxisbezogen schreibt und dabei aus seinen Erfahrungen als Mitbegründer des KDE schöpft, zeichnet sich der zweite Beitrag dieses Kapitels durch eine stark wissenschaftliche Herangehensweise aus. Der Autor, *Gregorio Robles*, forscht und unterrichtet an der Universidad Rey Juan Carlos in Madrid und befasst sich seit einigen Jahren intensiv mit dem Thema „Libre Software"[8]. Sein Artikel schildert die Bemühungen, das Phänomen mit den Methoden und Werkzeugen der Softwaretechnik zu erfassen und somit greifbarer und verständlicher zu machen. Die große Frage lautet: Wie und wann wird ein Projekt erfolgreich? Denn derzeit lassen sich keine verlässlichen Voraussagen über den Werdegang eines Projektes treffen, es ist stets ein Spiel mit vielen Unbekannten. Diese Unsicherheit soll durch eine softwaretechnische Herangehensweise reduziert werden. Das Problem dabei ist, dass viele Konzepte der Softwaretechnik auf die spezifischen Arten, wie Freie Software entsteht, nicht anwendbar sind. Ausgehend von dem bekannten Basar-Modell von Eric S. Raymond als einem charakteristischen Entwicklungsmodell für Freie Software,[9] zeigt Robles die Möglichkeiten und Grenzen der softwaretechnischen Modelle und Verfahren auf diesem neuen Terrain. In seinem Beitrag „A Software Engineering Approach to Libre Software" veranschaulicht er, wo die Softwaretechnik im Bereich der freien Softwareentwicklung derzeit steht und welchen Weg sie in der Zukunft einschlagen muss. Er stellt die wesentlichen Analysemöglichkeiten vor und erläutert anhand einiger bereits durchgeführter empirischer Studien, welche Erkenntnisse bisher gewonnen werden konnten.

Ein wesentlicher Grund für den Einzug der Softwaretechnik, also das systematische Vorgehen bei der Softwareentwicklung nach bestimmten Modellen, war der Wunsch, Software fehlerfreier und somit zuverlässiger und sicherer zu machen.

Da wie schon erwähnt die bekannten Verfahren der Softwaretechnik auf die Analyse quelloffener Software nicht so ohne weiteres anwendbar sind, könnte man

---

8 Libre Software wurde 1999 auf der IST-Konferenz in Helsinki als übergreifender Begriff für quelloffene Software vorgeschlagen, der sowohl Freie Software als auch Open-Source-Software abdecken sollte. Siehe dazu die Homepage der *European Working Group on Libre Software* unter http://eu.conecta.it/.

9 Robles betont aber, dass die Begriffe Freie-Software-Entwicklung und Basar-Modell nicht gleichzusetzen sind.

leicht zu dem Schluss kommen, Freie Software sei im Vergleich zu proprietär entwickelter vielleicht unsicherer, qualitativ schlechter. Dennoch werben Anbieter Freier Software häufig mit dem Schlagwort Sicherheit, die sie geradezu als einen entscheidenden Vorteil gegenüber der Konkurrenz aus dem Closed-Source-Lager propagieren. Und es sind in der Tat nicht wenige, die den quelloffenen Weg der Softwareentwicklung als den Erfolg versprechenderen in puncto Softwaresicherheit ansehen. Ein wesentlicher Grund für den Sicherheitsgewinn läge gerade in der Offenlegung des Programmcodes, denn sie ermögliche es Interessierten, seien es nun Anwender oder Entwickler,[10] den erstellten Programmcode zu überprüfen. Dadurch könnten Fehler schneller entdeckt und behoben werden. Eric S. Raymond drückte diesen Sachverhalt in seinem Artikel *The Cathedral and the Bazaar* mit folgendem viel zitierten Satz aus: „Given enough eyeballs, all bugs are shallow" (1999, S. 41).

Doch diese Argumentation stößt nicht überall auf Zustimmung. Ein beliebtes Gegenargument ist, dass, gerade weil sehr viele Hände am Programmcode herumwerkeln, die Qualität des Produktes leidet. Für die einen kann ein Projekt also nicht genug Beteiligte haben, die anderen betrachten Software wiederum als sprichwörtlichen Brei, den zu viele Köche bekanntlich verderben, und so ist seit einigen Jahren eine rege Debatte zum Thema Open Source und Softwaresicherheit in den Medien zu verfolgen.

Robert A. Gehring, wissenschaftlicher Mitarbeiter im Fachgebiet Informatik und Gesellschaft der TU Berlin, analysiert im dritten Beitrag dieses Kapitels die laufende Diskussion. Seine Arbeit mit dem Titel „Sicherheit mit Open Source – die Debatte im Kontext, die Argumente auf dem Prüfstein" versucht zunächst aufzuzeigen, welche Faktoren die Diskussion dominieren und in welchen Bereichen die Debatte nur unzureichend geführt wird bzw. welche Argumente schlichtweg übersehen oder übergangen werden. Ausgehend von dieser Analyse fokussiert Gehring die Frage nach dem Potenzial von Freier Software bezüglich Softwaresicherheit auf die Bereiche Softwaretechnologie und Software-Ökonomie. Er bezieht in seine Argumentation aktuelle, bislang in der Debatte wenig oder nicht berücksichtigte Arbeiten ein und bietet dem Leser mit dem vorliegenden Werk ein Dokument, das einen neuen Zugang zur Thematik erschließen hilft.

Bislang wurde hier die Betrachtung auf die Entstehung und, unter dem Aspekt der Softwaresicherheit, auf bestimmte Eigenschaften von Freier Software fokussiert. Will man dem Thema Open Source aus technischer Sicht einigermaßen gerecht werden, kommt man aber nicht umhin, den Blick zu heben und auch über das Umfeld von Freier Software schweifen zu lassen. Welche Einflüsse wirken auf die Open-Source-Bewegung ein? Welche Auswirkungen hat diese auf andere „Geschöpfe" der Informationstechnologie? Ein sehr beeindruckendes Geschöpf der Informationstechnologie ist das Internet. Dass das Internet mit seinen Kommunikationsmöglichkeiten entscheidend zum Erfolg der Open-Source-Bewegung beigetragen hat, dürfte spätestens nach der Lektüre des Beitrags von Matthias Ettrich besser klar werden. Doch auch Freie Software hat wiederum ihren Anteil an der Erfolgs-

[10] Die Grenze zwischen Anwender und Entwickler verschwimmt bzw. verschwindet im Open-Source-Umfeld, da der Anwender eigenhändig Verbesserungen an der Software vornehmen kann. Näheres dazu im Beitrag von Matthias Ettrich.

story des WWW. Wer kann sich das Internet ohne offene Protokolle wie TCP/IP oder freie Skriptsprachen wie PHP oder Perl vorstellen? Mit der voranschreitenden Vernetzung ist auch das Bedürfnis nach Interoperabilität gestiegen. Software soll nicht länger nur als eigenständiges Programm auf einem lokalen Rechner laufen, sondern sich als Baustein in ein System aus verteilten Diensten integrieren. Doch wie erreicht man es, aus einem Dschungel von unternehmensspezifischen Implementierungen ein funktionierendes Ganzes zu bilden? Man schafft Standards. Das für das Internet zuständige Standardisierungsgremium ist das *World Wide Web Consortium (W3C)*. Seit 1994 hat das aus rund 400 Organisationen bestehende Konsortium so namhafte Bestandteile des Internets wie das *Hypertext Transfer Protocol (HTTP)* oder die *Extensible Markup Language (XML)* durch seine Standardisierungsverfahren geschickt und mittlerweile bereits über 50 Spezifikationen erstellt.

Mit dem Beitrag von *Dirk Kuhlmann*, Forscher an den *Hewlett Packard Laboratories* in Bristol, riskieren wir einen Blick über den semantischen Tellerrand des Begriffes „Open Source" und beschäftigen uns mit der Welt der Standards, genau genommen der *offenen Standards*. Kuhlmann geht in seinem Artikel „Open Source und offene Standards" der grundlegenden Frage nach, was Offenheit überhaupt bedeutet. Während bei Open Source die Offenheit, wie der Name schon sagt, in der Verfügbarkeit des Quellcodes begründet liegt, fällt es bei Standards weniger leicht, Offenheit konkret festzumachen. Ausgehend von dieser Fragestellung führt er den Leser auf unkonventionelle Art durch die Wirren der Standardisierungsmechanismen und schafft es immer wieder, Brücken zu verwandten Themen wie den Softwarepatenten oder der Kontroverse um die *Trusted Computing Platform Alliance (TCPA)* zu schlagen. Die Open-Source-Bewegung lässt Kuhlmann dabei stets im Hintergrund schweben, um sie an geeigneter Stelle ins Bewusstsein des Lesers zu rücken und deren Rolle oder Potenzial in den Fokus zu stellen.

Damit endet der kleine Exkurs in die Welt der offenen Standards, und zum Ende des Kapitels wird noch einmal ein praxisrelevanter Aspekt des Themas ins Blickfeld gerückt. Nicht zuletzt die Migration der Münchener Stadtverwaltung auf Open-Source-Software hat gezeigt, dass ein wachsendes Interesse bei Behörden und Unternehmen zu beobachten ist, auf Open-Source-Produkte umzusteigen. Als Beweggründe sind hierbei u.a. der Wunsch, nicht von einem Softwarehersteller abhängig zu sein, die geringeren Kosten einer freien Softwarelösung sowie mangelndes Vertrauen (ob zu Recht oder nicht, sei dahingestellt) in die Sicherheit proprietärer Software genannt worden (SPD-Fraktion im Münchner Rathaus 2003). Vielleicht trägt sich der eine oder andere Leser ebenfalls gerade mit dem Gedanken, in seiner Firma Linux als Betriebssystem einzuführen. Dann dürfte der letzte Beitrag dieses Kapitels mit dem Titel „Erfolgsfaktoren bei der Einführung von Linux in Unternehmen" von *Peter Ganten* besonders interessieren. Der Aufsatz klärt über Schwierigkeiten und Risiken auf, die bei einer Einführung von Open-Source-Software zu beachten sind. Da er sich im Kapitel „Technik" befindet, stehen dabei nicht finanzielle, sondern technische und strukturelle Aspekte im Vordergrund. Dennoch ist klar, dass bei einem solch gewichtigen Unterfangen wie der Umstellung der IT-Infrastruktur einer Firma oder einer Behörde die Frage der Wirtschaftlichkeit nicht außen vor gelassen werden kann.

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

Als Gründer und Geschäftsführer eines Unternehmens, das sich auf die Implementierung von (bzw. Migration zu) Linux spezialisiert hat, kann Peter Ganten aus den Erfahrungen zahlreicher Projekte für Firmen und Behörden schöpfen. In seinem Beitrag beschreibt er, wie die Planung und Durchführung einer Umstellung der Server und Clients auf Linux bei einem mittelständischen Unternehmen aussehen könnten. Dazu schildert er zunächst die Entwicklung einer Open-Source-Strategie und darauf aufbauend die letztendliche Umsetzung der Migration.

Die vorgestellten Beiträge spannen einen Bogen von der Entstehung bis zum konkreten Einsatz quelloffener Software und sollten dem Leser somit einen umfassenden Überblick über die technische Seite von Open Source vermitteln können. Da von verschiedenen Seiten gerne Pauschalurteile über die Qualität und das Potenzial von Freier Software gefällt werden, ist eine umfassende Kenntnis der technischen Hintergründe äußerst wichtig. In diesem Sinne: Eine erhellende Lektüre!

Literatur

Brooks Jr., Frederick P. (1975): *The Mythical Man-Month: Essays on Software Engineering*, Addison Wesley

Bezroukov, Nikolai (1999): *Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)*, First Monday, volume 4, number 10 (October 1999), online http://www.firstmonday.dk/issues/issue4_10/bezroukov/ (25.1.2004)

González-Barahona, Jesús M. und Robles, Gregorio (2003): *Free Software Engineering A Field to Explore*, in: Upgrade Vol. IV, No. 4 (August 2003), S. 49–54, online http://www.upgrade-cepis.org/issues/2003/4/up4-4Gonzalez.pdf (25.1.2004)

heise online News, 30.8.2000: *Branchenschwergewichte investieren in Linux* online http://www.heise.de//newsticker/data/vza-30.08.00-000/ (25.1.2004)

Raymond, Eric S. (1999): *The Cathedral and the Bazaar*, S. 27–78, in: Eric S. Raymond: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Sebastopol, CA: O'Reilly, online http://www.openresources.com/documents/cathedral-bazaar/main.html (25.1.2004)

SAP (2003): *SAP stellt ihr Datenbankprodukt der weltweit populärsten Open Source Community zur Verfügung*, Pressemitteilung vom 26.5.2003, online http://www.sap.com/germany/aboutSAP/press/press_show.asp?ID=1081 (25.1.2004)

Sommerville, Ian (2001): *Software Engineering*, 6. Aufl., Pearson Studium, München

SPD-Fraktion im Münchner Rathaus (2003): *Rathaus-SPD entscheidet sich für Linux*, Presseerklärung vom 26.5.2003, online http://www.spd-rathaus-muenchen.de/presse/press39-linux_030526.pdf