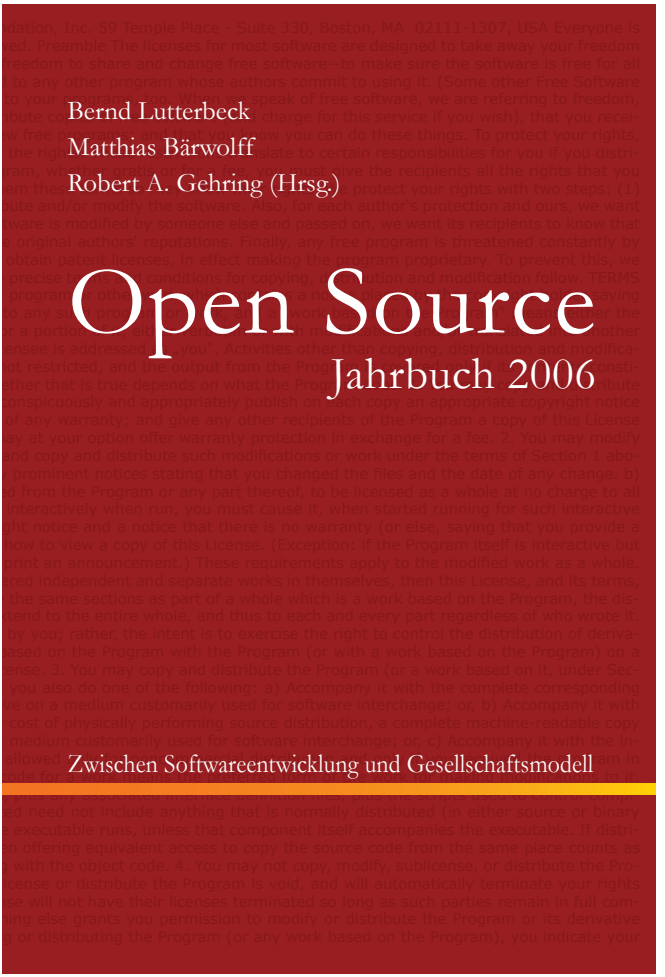
Dieser Artikel ist Teil des

Open Source Jahrbuchs 2006



erhältlich unter www.opensourcejahrbuch.de.

Die komplette Ausgabe enthält viele weitere interessante Artikel. Lob und Kritik zu diesem Artikel sowie weitere Anregungen können Sie uns einfach und unkompliziert mitteilen per E-Mail oder auf www.opensourcejahrbuch.de/feedback/.

Mythen, Märchen, Missverständnisse – Eine nüchterne informatische Betrachtung von Open-Source-Entwicklungsprozessen

MATTHIAS FINCK UND WOLF-GIDEON BLEEK

In der Literatur finden wir viele Vorurteile und Missverständnisse über Open–Source-Prozesse. Diese betreffen sowohl die Zusammensetzung und die Arbeitsweise der Beteiligten als auch deren Motivation, Qualifikation und Organisation. Eine ausführliche Zusammenstellung der aktuellen Literatur zu diesem Thema wird ausgebreitet und deren Widersprüche aufgezeigt. Durch eine von der Informatik geprägte Perspektive bringt dieser Artikel eine alternative Betrachtungsweise in das Feld der Open-Source-Entwicklung. Damit helfen wir ein neues – stärker informatisch geprägtes – Feld von Forschungsfragen zu erschließen und vorhandene unter einem neuen Licht zu erörtern.

Schlüsselwörter: Softwareentwicklung · Prototyp · Motivation · Eigenbedarf · Organisation

1 Einleitung

Über OS-Software[1] und die dahinter liegenden Entwicklungsprozesse ist in der vergangenen Zeit eine Menge geschrieben worden. Von weltweiten kollektiven Entwicklungsprozessen über jugendliche Hacker bis hin zu Anarchie. Unabhängig davon, auf welcher Basis und in welcher Euphorie diese Aussagen getroffen wurden, feststeht, dass OS-Prozesse nicht den bekannten Kategorien von Entwicklungsprozessen, bei denen als Unterschied lediglich das Produkt quelloffen vorliegt, entsprechen.

Die tatsächlichen Besonderheiten von OS-Entwicklungsprozessen zu identifizieren, ist aus zweierlei Hinsicht eine Herausforderung. Zum einen existiert bisher keine allgemein anerkannte Definition für OS-Entwicklungsprozesse, weil alle Prozesse

1 OS steht in diesem Beitrag als Abkürzung für Open Source.

zusammengenommen so unterschiedlich sind, dass sich die Prozessbeschreibung auf die Eigenschaften des Ergebnisses, den offenen Quellcode, reduziert. Zum anderen werden in existierenden Arbeiten zu OS die Eigenschaften häufig verklärt dargestellt, was die genaue Analyse der spezifischen Eigenschaften erschwert.

Um die Unterschiede zu Software-Entwicklungsprozessen im Allgemeinen aufzuarbeiten und sich dabei möglichst wenig von mit OS verbundenen Mythen oder Missverständnissen leiten zu lassen, betrachten wir OS-Entwicklungsprozesse in diesem Beitrag aus der uns eigenen informatischen Perspektive als Software-Entwicklungsprozesse mit ganz charakteristischen Merkmalen. Wir werden aus unserer Perspektive der Softwareentwicklung die Menge der OS-Entwicklungsprozesse soweit einschränken, dass wir *Offenheit*, *Verteiltheit* und *Agilität* als die drei charakteristischen Merkmale benennen können, die in ihrer Ausprägung bei OS-Entwicklungsprozessen die signifikanten Unterschiede zu anderen Entwicklungsprozessen darstellen.

Dazu beschreiben wir zunächst unsere Perspektive auf Softwareentwicklung und damit auf OS-Entwicklungsprozesse. Danach werden wir eine Reihe von aus unserer Sicht existierenden Mythen und Missverständnissen im Zusammenhang mit OS-Prozessen aufdecken, um uns dann aus der uns eigenen informatischen Perspektive den verbleibenden Eigenschaften zuzuwenden und diese zu analysieren. Abschließend stellen wir die aus unserer Sicht charakteristischen Merkmale von OS-Software vor.

2 Eine informatische Perspektive auf OS-Entwicklungsprozesse

OS-Entwicklungsprozesse werden aus den unterschiedlichsten Perspektiven heraus betrachtet. Lange Zeit dominierte eine ökonomische Betrachtung den Blick auf OS–Phänomene. Inzwischen existiert aber auch eine Reihe von juristischen, pädagogischen, politischen und sozialwissenschaftlichen Arbeiten zu diesem Thema.

Interessanterweise gibt es vergleichsweise wenig Arbeiten aus einer informatischen Perspektive, obwohl es sich hierbei doch um die Betrachtung von Software-Entwicklungsprozessen handelt. Diese Perspektive wollen wir in diesem Beitrag einnehmen.

Dabei nehmen wir in Bezug auf Softwareentwicklung eine anwendungsorientierte und menschenzentrierte Haltung ein (Floyd und Züllighoven 1999; Züllighoven 2004). Wir verstehen Softwareentwicklung als einen partizipativen Prozess, der die Perspektiven der beteiligten Personen einbezieht und diese an den Entwicklungsentscheidungen teilhaben lässt (Floyd und Züllighoven 1999; Schuler und Namioka 1993). Wir tragen damit dem Umstand Rechnung, dass Softwarequalität in der Anwendung eine subjektive Eigenschaft ist, die neben etablierten Standards eine Aushandlungskomponente besitzt, die in einem dialogischen Prozess am besten hergestellt werden kann. Anwendungsorientierte Softwareentwicklung verstehen wir weiterhin als einen Prozess, der durch die in kurzen Zyklen entwickelten und eingesetzten Prototypen versucht, durch eine enge Verzahnung von Problemidentifikation, Lösungserarbeitung, technische Umsetzung, der Anwendung des Prototypen im Einsatzkontext und

der Evaluation der Anwendung eine Konvergenz hin zu einem Softwareprodukt zu erzielen (Budde et al. 1992; Bleek et al. 2004).

Wir halten für unseren Beitrag diese Perspektive für zielführend, da es sich bei der Entwicklung von OS-Software eben auch um einen Software-Entwicklungsprozess handelt. Aus diesem Grund werden wir uns in diesem Beitrag nur auf Mythen bzw. Missverständnisse im Zusammenhang mit OS konzentrieren, die in unmittelbarem Zusammenhang des Entwicklungsprozesses stehen. Zu hinterfragende Aussagen über z. B. die wirtschaftlichen Aspekte von OS-Software bleiben hier unberücksichtigt.

3 Mythen, Märchen und Missverständnisse – ein Überblick

Mit OS-Software werden im Moment viele Wünsche und Chancen verbunden. Dies reicht von dem Wunsch, Software für alle zu entwickeln – vor allem für diejenigen, die sich teure Lizenzen nicht leisten können – über die Prozesse der OS-Entwicklung als Prototyp neuer gesellschaftlicher Bewegungen bis hin zu den Chancen der Erschließung neuer Märkte über Dienstleistungen im Zusammenhang von OS. Diese Chancen und Wünsche sehen wir ebenfalls und wollen sie in keiner Weise schmälern.

Dass OS-Prozesse doch ganz spezifische Merkmale aufweisen, die es lohnt zu betrachten, darauf weist z. B. O'Reilly (2005, S. 463) in seinem Beitrag „Open Source Paradigm Shift" eindrücklich hin. Er sieht in Open-Source-Entwicklungsprozessen eine Innovation im Hinblick auf drei tief greifende, lang währende Trends in der Softwareentwicklung: die Entwicklung von Software zum Massenprodukt, die Netzwerk-gestützte Zusammenarbeit und die Anpassbarkeit von Software an Kundenbedürfnisse (Software als Dienstleistung). Er macht in diesem Zusammenhang aber unmissverständlich klar, dass, um die Innovation und ihre Auswirkungen auf Entwicklungsprozesse zu verstehen, nicht die Definition von offenem Quellcode als Erklärung ausreicht, sondern, dass die spezifischen Besonderheiten von typischen OS-Prozessen erkannt und untersucht werden müssen.

> „We must understand how the means by which software is deployed changes the way in which it is created and used. We must also see how the same principles that led to early source code sharing may affect other fields of collaboration activity. Only when we stop measuring open source by what activities are excluded from the definition and begin to study its fellow travellers on the road to the future will we understand its true impact and be fully prepared to embrace the new paradigm." (O'Reilly 2005, S. 480)

Um diese Besonderheiten identifizieren zu können, ist uns daran gelegen, die Wunschvorstellungen von der Realität zu differenzieren und die momentan existierenden OS-Prozesse aus informatischer Perspektive wissenschaftlich möglichst detailliert zu untersuchen, um die mit OS verbundenen Visionen besser beurteilen zu können.

3.1 Die Beteiligten und ihre Motivation

Hartnäckig hält sich die Vorstellung von 15- bis 25-jährigen, jugendlichen Hackern, die aus Abneigung gegen Bill Gates und andere Software-Giganten sich zusammenschließen und gemeinsam gegen die kommerzielle Macht antreten und Software für eine bessere Welt entwickeln. Diese verbreitete Betrachtung der Beteiligten ist auf zweierlei Weise verkürzt. Zum einen handelt es sich bei den EntwicklerInnen, nicht (nur) um die beschriebene Gruppe von Personen, zum anderen existieren neben ihnen noch eine Reihe weiterer Personen, die meist nicht mit betrachtet werden, obwohl sie wesentlichen Einfluss auf den Entwicklungsprozess haben.

Aus unserer beschriebenen Perspektive genügt es bei einer Betrachtung der Prozesse nicht, sich nur auf die EntwicklerInnen als Beteiligte zu konzentrieren. Sowohl die NutzerInnen der Software als auch am Prozess beteiligte Organisationen machen einen wichtigen Teil der Interessensgruppen aus. Gerade die Gruppe der AnwenderInnen, die ein OS-Produkt letztlich wie proprietäre Software verwenden, findet jedoch in den bisherigen Betrachtungen kaum Berücksichtigung. Wir werden deshalb die beteiligten Interessengruppen benennen und deren Motivation, sich in dem Prozess einzubringen, analysieren.

Die Beteiligten sind jugendliche Hacker

Feller und Fitzgerald (2001) zufolge sind OS-EntwicklerInnen in großer Prozentzahl keine Amateure, sondern Experten im Bereich Softwareentwicklung mit einer entsprechenden Ausbildung. Robles et al. (2001) haben ermittelt, dass 80 % der OS-EntwicklerInnen im IT-Bereich tätig sind. Eine umfassende Studie des Linux-Kernel-Projekts von Hertel et al. (2003) lässt sich auf andere OS-Projekte übertragen und besagt, dass 96 % der EntwicklerInnen männlich (womit dieses Vorurteil bestätigt scheint) und 70 % Prozent der Beteiligten zwischen 20 und 39 Jahre alt sind. Dies zeigt, dass es sich bei der Mehrzahl der EntwicklerInnen um IT-Fachleute handelt, die weitestgehend am Anfang ihrer Karriere stehen.

Abgesehen von der Fehleinschätzung der an der direkten Implementierung Beteiligten, werden in der Regel die übrigen in den Prozess involvierten Personen außer Acht gelassen. Weitere Gruppen von Interessenvertretern sind die Nutzenden und beteiligte Organisationen. Die Nutzenden werden dabei oftmals mit den EntwicklerInnen gleichgesetzt. Durch die Offenheit des Codes wird ein Bild beschrieben, wonach Erstere sich über die Nutzung der Software und Feedback zu der Nutzung in den Prozess einbringen. Sie werden so nach und nach zu EntwicklerInnen der Software. Dabei wird ein Interesse der Nutzenden an der Gestaltung und ihre Befähigung zur Implementierung vorausgesetzt. Das mag vor allem daran liegen, dass lange Zeit OS-Software hauptsächlich außerhalb der Anwendungssoftware erfolgreich eingesetzt wurde (z. B. Apache-Webserver, Linux-Kernel). Durch die zunehmende Marktdurchdringung von OS-Software auf dem Gebiet der Anwendungssoftware (z. B. Firefox,

OpenOffice, Gimp) wächst die Gruppe von Nutzenden schnell an und nimmt eine relevante Stellung ein. Und unter ihnen ein immer größerer Anteil „reiner" AnwenderInnen; sprich ein Personenkreis, der weder über die notwendige Qualifikation noch über die Motivation verfügt, sich über die reine Anwendung hinaus mit der Software zu beschäftigen. Darüber hinaus sind in den meisten OS-Projekten ebenfalls Organisationen beteiligt.

Weltverbesserung als Leitmotiv

Es bleibt die Frage nach dem Motiv, aus dem heraus die Beteiligten handeln. Weit verbreitet scheint die Vorstellung, OS-EntwicklerInnen partizipierten in den Projekten, um die Welt zu verbessern und den Monopolisten ein Bein zu stellen. Diese Motive sind mit Sicherheit mit ausschlaggebend für die Entstehung der Bewegung gewesen. Aber gilt dies auch heute noch? Weber (2004, S. 9) stellt bei der Betrachtung von OS-Prozessen Folgendes fest:

> „Intuition tells us that thousands of volunteers are unlikely to come together to collaborate on a complex economic project, sustain that collaboration over time, and build something that they give away freely, particularly something that can be beat some of the largest and richest business enterprises in the world at their own game."

Er trifft diese Aussage in Anspielung auf die Gruppe von Personen, die an dem Betriebssystem Linux und dessen Anwendungsprogrammen (z. B. OpenOffice, Gimp) mitentwickeln. Das scheinbare Paradoxon verleitet zu einer genaueren Analyse der am Prozess beteiligten Personen und deren Motivation.

Raymond (1999) beantwortet die von Lerner und Tirole (2002) aufgeworfene Frage, warum sich tausende EntwicklerInnen freiwillig an der Bereitstellung eines öffentlichen Gutes beteiligen sollten, mit dem Hinweis auf drei grundlegende Motive, sich an einem OS-Prozess zu beteiligen: Erstens ziehen die Beteiligten einen direkten Vorteil aus der Software, die sie mitentwickelt haben, da sie sie in der Regel selbst verwenden, zweitens lieben die meisten von ihnen die Arbeit als ProgrammierIn an sich und drittens können sie ihre Reputation als ProgrammiererIn erhöhen, indem sie einen hochqualifizierten Beitrag zu einem OS-Projekt geleistet haben.

Diese drei von Raymond (1999) aufgestellten Mutmaßungen über die Motivation von OS-EntwicklerInnen, die er aus seiner langjährigen Erfahrung als aktives Mitglied der OS-Bewegung heraus gemacht hat, sind in zahlreichen empirischen Studien (Robles et al. 2001; Jørgensen 2001; Hars und Ou 2001; Ghosh et al. 2002; Lakhani und Wolf 2001; Lakhani und von Hippel 2003) und Sekundäranalysen (Luthiger 2004; Weber 2004) bestätigt, ausdifferenziert und ergänzt worden. Demnach sind die Hauptmotive *Eigenbedarf*, *Spaß*, *Lernprozess*, *Reputation*, *Bezahlung*, *Identifikation*, *Altruismus*.

Die verschiedenen Untersuchungen zeigen jedoch, dass die altruistischen Motive bei OS-EntwicklerInnen lange nicht so dominant sind, wie häufig behauptet wird. Auf

der anderen Seite wird der finanzielle Aspekt bei der Diskussion um die Motivation der OS-EntwicklerInnen vielfach unterschätzt: Circa die Hälfte der EntwicklerInnen wird für ihr Engagement in einem OS-Projekt zumindest teilweise bezahlt. Der Wunsch, etwas zu lernen und dadurch Wissen und Reputation zu erhöhen bzw. Spaß bei der Entwicklung zu haben, nimmt bei allen in dieser Arbeit vorgestellten Befragungen eine deutlich stärkere Gewichtung ein. Das Hauptargument an der Entwicklung ist der Eigenbedarf.

Unabhängigkeit und Überparteilichkeit

Eine weitere weit verbreitete Vorstellung von OS-Entwicklungsprozessen ist die, dass OS-Software als Gegengewicht zu proprietärer Software entwickelt wird und die Unabhängigkeit von großen Softwarefirmen gewährleisten soll. Tatsache ist jedoch, dass viele der erfolgreichen OS-Produkte stark vom Engagement großer Firmen abhängig sind. So wird z. B. OpenOffice von der Firma SUN entwickelt.

Feller und Fitzgerald (2001) betrachten mit IBM, Red Hat und Cosource drei große Organisationen, die ein kommerzielles Interesse an OS-Software haben und entwickeln auf der Basis dieser Beobachtungen Eindrücke von gemeinsamen Eigenschaften solcher Firmen, die ein kommerzielles Interesse mit ihrem Engagement in OS-Projekten verbinden. Diese Firmen vertreten ihre eigenen Interessen in den OS-Entwicklungsprozessen. Und diese liegen in der Erschließung neuer Märkte und Dienstleitungen durch OS-Software. Aus dem Grund nehmen die kommerziellen Organisationen im Umfeld von OS-Entwicklungsprozessen in zwei unterschiedlichen Rollen Einfluss auf OS-Entwicklungen. Auf der einen Seite als Kunde, da sie die Software benötigen, um ihrerseits ihr Klientel bedienen zu können und auf der anderen Seite als „Gönner der Entwicklung" (Feller und Fitzgerald 2001, S. 121), weil sie die Entwicklung fördern bzw. beeinflussen, indem sie Ressourcen zu Verfügung stellen, um den Entwicklungsprozess in eine von ihnen gewünschte Richtung voranzutreiben. Das bedeutet gerade für kleinere Projekte, dass sie keineswegs unabhängig sind, sondern maßgeblich durch die beteiligten Unternehmen zumindest indirekt gesteuert werden, weil durch die kommerziellen Organisationen finanzielle Mittel und somit auch Arbeitskraft zur Verfügung gestellt werden, die in der Regel an bestimmte der Organisation zuträglichen Entwicklungsarbeiten geknüpft sind.

3.2 Der Prozess und dessen Organisation

Die Grenzenlose Offenheit

Bei differenzierter Betrachtung lässt sich schnell feststellen, dass die Offenheit in OS-Entwicklungsprozessen Grenzen hat. Zum einen wird immer behauptet, durch die Verfügbarkeit des Quellcodes steht ein OS-Projekt allen Menschen offen. Dies stimmt in den meisten OS-Projekten nur für diejenigen, die in der Lage sind, sich an der Implementierung des Quelltextes zu beteiligen. Allen anderen, wie z. B. den

reinen AnwenderInnen, erscheinen OS-Projekte in keiner Weise offener als Projekte zur Entwicklung proprietärer Software.

Außerdem bedeutet Offenheit nicht, dass jeder Person alle Möglichkeiten, sich in dem Projekt einzubringen, offen stehen. In der Regel existieren verschiedene Rollen in OS-Projekten, die mit unterschiedlichen Kompetenzen verbunden und in unterschiedlichem Maße offen sind (Elliott und Scacchi 2005). Möchte jemand Teil der Entwicklungsgemeinschaft werden, muss diese Person in der Regel abgestuft verschiedene Nachweise erbringen, um entsprechende zusätzliche Kompetenzen zu erwerben. So werden EinsteigerInnen meistens erst einmal mit der Behebung von (einfachen) Fehlern betraut, bevor sie sich an der Weiterentwicklung beteiligen dürfen.

Am Unterschied des Zugriffs auf den Code werden ebenfalls die Grenzen der Offenheit deutlich. Jede Person darf lesend auf den Quelltext eines OS-Projekts zugreifen. Die Möglichkeit, durchgeführte Änderungen dann in den gemeinsamen Code einfließen zu lassen, ist begrenzt, da in der Regel nur ein kleiner Kreis über die Rechte des schreibenden Zugriffs auf den gemeinsamen Code verfügt.

Alles in allem lässt sich die Offenheit eines OS-Projekts als dreigestuft begreifen:

- Nutzung der OS-Software: Die Software steht jeder Person frei zur Verfügung.
- Lesender Zugriff auf den Code: Der Quellcode und das Recht, Veränderungen am Quellcode vorzunehmen stehen den Personen frei, die über entsprechende Kompetenzen verfügen.
- Schreibender Zugriff auf den Code: Die letztliche Entscheidungsgewalt darüber, was in den Code zurückfließt, besitzen in der Regel nur wenige, erfahrene, aktive und seit langer Zeit dem Projekt zugehörige EntwicklerInnen, die über ein schreibenden Zugriff verfügen.

Die Selbstorganisation der OS-Projekte

Durch die Offenheit und die damit verbundene Freiwilligkeit bezüglich der Partizipation werden OS-Projekte als selbstorganisierte Projekte bezeichnet. Autoren wie z. B. Weber (2004) warnen allerdings nachdrücklich davor, im Zusammenhang mit OS-Projekten von *Selbstorganisation* zu sprechen. Zum einen wird diese für ihn zu oft als Platzhalter für einen unergründeten Mechanismus verwendet und somit als Euphemismus für fehlendes Verständnis der Zusammenhänge, die die Organisation tatsächlich zusammenhalten.

> „[S]elf-organization is used too often as a placeholder for an unspecific mechanism. The term becomes a euphemism for: I don't really understand the mechanism that holds the system together." (Weber 2004, S. 132)

Der Begriff *Selbstorganisation* wird in diesem Zusammenhang definiert, indem er als Gegensatz zu umfassender Autorität als eine Organisation verstanden wird, in der

die Anweisungen unmittelbar aus dem lokalen Handeln kooperierender Individuen heraus entstehen. Weber führt zum anderen an, dass selbst wenn Selbstorganisation als Begriff definiert wird, dies die Vorgänge nur unzureichend und nicht zutreffend umschreibt.

Auch wird mit dem Begriff der Selbstorganisation die Organisationsform von OS-Projekten idealisiert. Sie wird als angestrebter Zustand seitens aller Beteiligten angesehen, obwohl die so genannte Selbstorganisation – vielleicht im Unterschied zu den Anfängen der OS-Bewegung – heute in vielen Projekten kein explizit angestrebtes Ziel ist, sondern eine Notwendigkeit, die sich aus der Offenheit der Projekte und der daraus resultierenden Freiwilligkeit der beteiligten Personen ergibt.

Statt Selbstorganisation ließe sich dieser Zustand auch mit informeller Planung und geringer Kontrolle umschreiben. Aufgrund fehlender Sanktionsmöglichkeiten existieren in OS-Projekten kaum feste Zeitzusagen oder planmäßige Absprachen (Persson et al. 2005). Stattdessen wird das allgemeine Langzeitziel formuliert, das Produkt verbessern zu wollen (Johnson 2001, S. 77). Außerdem existieren inzwischen eine Reihe von OS-Projekten, die z. B. durch eine Firma mit kommerziellen Interessen finanziell unterstützt werden, so dass die Firma einen erheblichen Einfluss auf den Entwicklungsprozess entwickelt. Johnson (2001) unterscheidet im Zusammenhang mit OS-Entwicklungsprozessen drei Varianten dezentraler Zusammenarbeit: die demokratische Variante, die darauf basiert, dass alle Beteiligten „selbstlos" zum Allgemeinwohl mitentwickeln; die zentral kontrollierte Variante, bei der ein zentrales Leitungsgremium die Entscheidungen trifft, und drittens die kontrolliert dezentrale Variante, die als Kompromiss die Stärken beider vorheriger Möglichkeiten versucht zu vereinen.

> „The team has a leader who coordinates specific tasks and secondary leaders that have responsibility for defined subtasks. Problem solving remains a group activity, but implementation is partitioned among subgroups." (Johnson 2001, S. 56)

Während im Zusammenhang mit OS-Entwicklungsprozessen häufig von der rein demokratischen Variante als Idealbild der Selbstorganisation ausgegangen wird, sieht Johnson die Form der kontrolliert dezentralen Zusammenarbeit als typisch für OS–Prozesse an. Auf der einen Seite wollen und werden die beteiligten EntwicklerInnen aufgrund der Freiwilligkeit ihres Engagements im hohen Maße selbst darüber entscheiden, wie sie was umsetzen. Auf der anderen Seite müssen die individuellen Wünsche und Ziele koordiniert werden, um die Gesamtentwicklung nicht zu gefährden.

Größe und Verteiltheit

OS-Projekte werden als prototypische Beispiele weltweit verteilter Entwicklungsprozesse angesehen. In der Tat gibt es eine Reihe prominenter Projekte, für die diese

Beschreibung zutrifft. Doch dieses Merkmal als notwendiges für eine OS-Entwicklung anzusehen, hieße, die Masse der Entwicklungsprozesse nicht zu betrachten. Feller und Fitzgerald (2001) beschreiben die weltweite Verteilung als Kriterium für große OS-Entwicklungsprozesse. Eine analytische Betrachtung jedoch schränkte den Begriff weltweit stark ein. Auch wenn in großen Projekten wie Linux Personen aus mehr als 30 Ländern an der Entwicklung in irgendeiner Form beteiligt waren oder sind (Weber 2004, S. 69), so stammen diese Personen zu über 90 % aus Europa, Nordamerika oder Australien. Werden die Untersuchungsergebnisse hinzugezogen, dass ca. Dreiviertel des beigetragenen Codes von maximal 10 % der Beteiligten stammt (Weber 2004, S. 71), so relativiert sich die Aussage der weltweiten Entwicklung von Open-Source-Entwicklungsprozessen.

An der weltweiten Verteiltheit von OS-Projekten lässt sich weiter zweifeln, werden Zahlen berücksichtigt, wonach in den 100 erfolgreichsten OS-Projekten durchschnittlich 6,6 Personen beteiligt sind (Krishnamurthy 2002). Die Hauptarbeit vieler OS–Projekte wird also von wenigen Personen getragen, die zumeist auch nicht weltweit verteilt arbeiten.

Entwicklungsgeschwindigkeit

Feller und Fitzgerald (2001) und andere bewerten gerade die o. g. Verteiltheit als Vorteil, da sie z. B. den Prozess beschleunigen soll. Die Masse an Menschen und die durch die Verteiltheit entstehende Möglichkeit, 24 Stunden an einem Produkt zu arbeiten, sollen im erheblichen Maße zu einem Geschwindigkeitsvorteil von OS-Projekten führen. Was neben der tatsächlichen Größe dabei außer Acht gelassen wird, ist der immense Koordinationsaufwand in OS-Prozessen.

Bedingt durch die Verteiltheit findet Kommunikation in OS-Entwicklungsprozessen überwiegend asynchron statt, denn wenn nicht durch unterschiedliche Zeitzonen voneinander getrennt, so arbeiten die Beteiligten aufgrund anderer Verteiltheitsaspekte, wie z. B. unterschiedlicher Arbeitszeiten, zu verschiedenen Zeitpunkten an der Software, was die Entscheidungsfindung erheblich verlangsamt.

Außerdem müssen diese Entscheidungen aufgrund der geringen Managementstrukturen und der damit verbundenen geringen Kontrolle auf die Handlung meist im Konsens getroffen werden. Ein zu geringes Maß an Mitsprache und Transparenz im Entscheidungsprozess führt schnell dazu, dass diejenigen, die nicht beteiligt wurden, sich übergangen fühlen und die Entscheidung untergraben (Jensen und Scacchi 2004, S. 50). Durch einen stark demokratischen Entscheidungsprozess entstehen viele Konfliktsituationen, die möglichst konsensual gelöst werden müssen. Die Konflikte in OS-Projekten werden in der Regel ohne formelle Management-Techniken gelöst (Elliott und Scacchi 2005, S. 167), was sehr zeitaufwendig sein kann.

4 Was übrig bleibt: Offenheit, Verteiltheit, Agilität

Die Kritik an den „Vorurteilen" gegenüber OS-Entwicklungsprozessen soll nicht das Potential dieser Form der Softwareentwicklung in Frage stellen. Ganz im Gegenteil. Die nüchterne Betrachtung auf die am Prozess Beteiligten verstärkt diese eher noch. Zu sehen, dass nicht nur jugendliche Hacker an der Entwicklung beteiligt sind, sondern dass sich sowohl professionelle SoftwareentwicklerInnen als auch große Firmen in OS-Entwicklungsprozessen einbringen, unterstreicht das Potential und die Wertschätzung, die diesen Prozessen inzwischen beigemessen werden. Die Erkenntnis, dass die Motivation, sich an OS-Entwicklungsprozessen zu beteiligen, nicht nur auf Spaß oder altruistischen Motiven beruht, sondern ebenso zur Steigerung der Reputation dient bzw. im zunehmenden Maße bezahlt wird, spricht für OS-Entwicklung als ein tragfähiges Entwicklungsmodell.

Was die kritisch betrachteten Aspekte der Prozesseigenschaften anbelangt, so ist zu betonen, dass allen OS-Entwicklungsprozessen das Potential gemein ist, diese Eigenschaften zu erfüllen. Die Besonderheit von OS-Prozessen besteht mit Sicherheit darin, dass sie eine hohe Dynamik entwickeln und an beträchtlicher Größe und Verteiltheit gewinnen können. Viel entscheidender, als OS-Entwicklungsprozesse anhand dieser möglichen Konsequenzen wie z. B. weltweiter Verteiltheit zu beurteilen, scheint es für uns zu sein, die Ursachen, die solche Effekte ermöglichen, als charakteristische Merkmale für OS-Entwicklungsprozesse zu identifizieren.

Aus unser Perspektive heraus besitzen OS-Entwicklungsprozesse drei zentrale Eigenschaften, die in ihrer Kombination die Potentiale und Innovationskraft dieser Prozessform erklären: *Offenheit*, *Verteiltheit* und *Agilität*.

- Die Offenheit und die eng damit verbundene Freiwilligkeit in der Zusammenarbeit stellen die Grundvoraussetzung für die Entfaltung der spezifischen Eigenschaften von OS-Prozessen dar. Die Offenheit als Kerneigenschaft ergibt sich dabei unmittelbar aus der Definition von OS-Software. Deren Quellcode ist immer frei zugänglich, und somit ist jede Person (zumindest theoretisch) in der Lage, sich in den Entwicklungsprozess einzubringen.
- Die Verteiltheit ergibt sich zwangsläufig aus der Offenheit des Projekts, wobei das nicht automatisch eine örtliche, gar weltweite Verteiltheit impliziert. Die Verteiltheit kann ebenso zeitlicher oder organisatorischer Natur sein.
- Die Agilität im Sinne agiler Softwareentwicklung (Quellen) ergibt sich ebenfalls aus der Offenheit und beschreibt das Potential von OS-Entwicklungsprozessen, in kurzen Entwicklungszyklen die Software getrieben von Einzelinteressen weiterzuentwickeln.

Diese drei Eigenschaften können zu einer weltweiten Verteiltheit einer großen, weitestgehend „selbstorganisierten" Gemeinschaft von EntwicklerInnen führen, die die Software in einer hohen Dynamik weiterentwickelt.

Das Abschnitt 3 einleitende Zitat von O'Reilly (2005, S. 480) wieder aufnehmend, sind aus unserer softwaretechnischen Sicht in Zukunft vor allem zwei Dinge zu leisten:

- Basierend auf der Annahme, dass es sich bei den Eigenschaften Offenheit, Verteiltheit und Agilität um die drei zentralen Merkmale von OS-Entwicklungsprozessen handelt, ist es notwendig, diese drei Eigenschaften detaillierter zu beschreiben.
- Außerdem ist zu untersuchen, welche Maßnahmen – wie z. B. die Herstellung von Transparenz im Entwicklungsprozess – in der Organisation von Software–Entwicklungsprozessen dazu führen, dass sich aus den drei grundlegenden Eigenschaften die erhofften und vielfach zitierten Konsequenzen in Bezug auf einen OS-Entwicklungsprozess entwickeln.

Literatur

Bleek, W.-G., Jeenicke, M. und Klischewski, R. (2004), 'e-Prototyping: Iterative Analysis of Web User Requirements', *Journal of Web Engineering* **3**(2), S. 77–94.

Budde, R., Kautz, K., Kuhlenkamp, K. und Züllighoven, H. (1992), *Prototyping: An Approach to Evolutionary System Development*, Springer, Berlin.

Elliott, M. S. und Scacchi, W. (2005), Free Software Development: Cooperation and Conflict in a Virtual Organizational Culture, *in* S. Koch (Hrsg.), 'Free/Open Source Software Development', Idea Group, Hershey, PA, USA, S. 152–172.

Feller, J. und Fitzgerald, B. (2001), *Understanding Open Source Software Development*, Addison Wesley Professional, Berlin.

Floyd, C. und Züllighoven, H. (1999), Softwaretechnik, *in* P. Rechenberg und G. Pomberge (Hrsg.), 'Informatik-Handbuch', Carl Hanser Verlag, München, S. 771–798.

Ghosh, R. A., Glott, R., Krieger, B. und Robles, G. (2002), 'Free/Libre and Open Source Software: Survey and Study FLOSS Deliverable D18: Final Report Part IV: Survey of Developers'. http://www.infonomics.nl/FLOSS/report/Final4.htm [22. Jan 2006].

Hars, A. und Ou, S. (2001), Working for Free? Motivations of Participating in Open Source Projects, *in* 'Proceedings of the 34th Annual Hawaii International Conference on System Science', S. 253–350. http://csdl.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf [03. Feb 06].

Hertel, G., Niedner, S. und Herrmann, S. (2003), 'Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel', *Research Policy* **32**(7), S. 1159–1177.

Jensen, C. und Scacchi, W. (2004), Collaboration, Leadership, Control, and Conflict Negotiation in the Netbeans.org Community, *in* J. Feller, B. Fitzgerald, S. Hissam und K. Lakhani (Hrsg.), 'Collaboration, Conflict and Control – Proceedings of the 4th Workshop on Open Source Software Engineering', S. 48–52. http://opensource.ucc.ie/icse2004/Workshop_on_OSS_Engineering_2004.pdf [18. Jan 06].

Johnson, K. (2001), A Descriptive Process Model for Open-Source Software Development, Master's thesis, University of Calgary, Department of Computer Science. http://sern.ucalgary.ca/students/theses/KimJohnson/thesis.htm [19. Jan 2006].

Jørgensen, N. (2001), 'Putting It All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project', *Information Systems Journal* **11**(4), S. 321.

Krishnamurthy, S. (2002), 'Cave or Community? An Empirical Examination of 100 Mature Open Source Projects', *First Monday* **7**(6). http://firstmonday.org/issues/issue7_6/krishnamurthy/index.html [19. Jan 2006].

Lakhani, K. R. und Wolf, R. G. (2001), Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects, *in* J. Feller, B. Fitzgerald, S. Hissam und K. Lakhani (Hrsg.), 'Perspectives on Free and Open Source Software', MIT Press, Cambridge. http://freesoftware.mit.edu/papers/lakhaniwolf.pdf [19. Jan 2006].

Lakhani, K. R. und von Hippel, E. (2003), 'How Open Source Software Works: "Free" User-to-User Assistance', *Research Policy* **32**(6), S. 923–943. http://opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf [19. Jan 2006].

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), S. 197–234. http://turingmachine.org/opensource/papers/lerner2002.pdf.

Luthiger, B. (2004), Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 93–106.

O'Reilly, T. (2005), The Open Source Paradigm Shift, *in* J. Feller, B. Fitzgerald, S. Hissam und K. Lakhani (Hrsg.), 'Perspectives on Free and Open Source Software', MIT Press, Cambridge, S. 461–482. http://tim.oreilly.com/articles/paradigmshift_0504.html [19. Jan 2006].

Persson, A., Lings, B., Lundell, B., Mattsson, A. und Ärlig, U. (2005), Communication, Coordination and Control in Distributed Development: An OSS Case Study, *in* J. Feller, B. Fitzgerald, S. Hissam und K. Lakhani (Hrsg.), 'OSS 2005 – Proceedings of the First International Conference on Open Source Systems', S. 88–92. http://oss2005.case.unibz.it/Papers/65.pdf [19. Jan 06].

Raymond, E. S. (1999), The Revenge of the Hackers, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources – Voices from the Open Source Revolution', O'Reilly, Cambridge, MA, USA, S. 207–220.

Robles, G., Scheider, H., Tretkowski, I. und Weber, N. (2001), Who is doing it? A Research on Libre Software Developers, Projektarbeit, Lehrstuhl für Informatik und Gesellschaft, Technische Universität Berlin, Berlin. http://ig.cs.tu-berlin.de/lehre/s2001/ir2/ergebnisse/OSE-study.pdf [19. Jan 2006].

Schuler, D. und Namioka, A. (1993), *Participatory Design: Principles and Practices*, Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, MA, USA.

Züllighoven, H. (2004), *Object-Oriented Construction Handbook*, dpunkt.verlag, Heidelberg.