Alles aus Spaß?
Zur Motivation von Open-Source-Entwicklern

Benno Luthiger

Ein PC-Benutzer geht normalerweise davon aus, dass gute Software wertvoll ist und deshalb etwas kostet. Es ist nachvollziehbar, dass solche Benutzer erstaunt sind, wenn sie das erste Mal mit Open-Source-Software in Kontakt kommen, erst recht, wenn die Software von überzeugender Qualität ist. Wie kommt es, dass Personen solche Software schreiben und freigeben, wenn sie mit dieser Software viel Geld verdienen könnten?

In diesem Artikel wird im ersten Kapitel gezeigt, wie das Paradox des Open-Source-Phänomens aufgelöst werden kann, indem die geeignete Perspektive eingenommen wird. Nicht die Softwarefirmen, sondern die einzelnen Softwareentwickler sind die Aktoren, welche die Existenz von Open-Source-Software erklärbar machen.

Im folgenden Kapitel wird dargelegt, dass eine ganze Reihe von Motivationen die Open-Source-Entwickler zu ihrer Tätigkeit anspornen. Die immer zahlreicheren empirischen Untersuchungen zu Open-Source-Software bestätigen die Existenz dieser Motivationen weitgehend. Was allerdings noch fehlt, sind Abschätzungen über die Relevanz dieser verschiedenen Beweggründe.

Im dritten Kapitel wird das Forschungsprojekt „Fun and Software Development" (FASD) vorgestellt. Dieses Forschungsprojekt ist Teil der Promotion des Autors am Institut für betriebswirtschaftliche Forschung der Universität Zürich. Ziel der FASD-Studie ist es, den Anteil, den Spaß bei der Entstehung von Open-Source-Software spielt, möglichst genau zu bestimmen. Zu diesem Zweck werden mit einem Online-Fragebogen mindestens 2000 Open-Source-Entwickler sowie ca. 1000 Entwickler in kommerziellen Softwarefirmen befragt. Die Umfrage wird im ersten Halbjahr 2004 durchgeführt. Erste Ergebnisse werden Ende 2004 erwartet.

1. Benutzer-Programmierer als Open-Source-Aktoren

Mit Software kann man viel Geld verdienen. Sowohl der Einsatz wie auch die Erzeugung von Software machen erhebliche Gewinne möglich. Dazu kommt, dass Computer und softwaregesteuerte Apparate den Alltag, vor allem den geschäftlichen, durchdringen. Diese Erfahrungen machen es naheliegend, Phänomene im Zusammenhang mit Software in erster Linie durch die wirtschaftliche Brille zu betrachten.

In einer solchen wirtschaftlich geprägten Sichtweise erscheint das Open-Source-Phänomen offensichtlich als Paradox: Warum engagieren sich Leute, um ein qualitativ hochstehendes Produkt zu erzeugen, wenn dieses in der Folge frei zur Verfü-

gung gestellt wird? Warum verschenkt die eine Person ein wertvolles Gut, wo doch die andere damit viel Geld verdient?

Eine Möglichkeit, dieses Paradox aufzulösen, ergibt sich aus der Verschiebung des Blickwinkels. Statt, wie in der ökonomisch geprägten Sichtweise naheliegend, die Softwarefirmen als die relevanten Aktoren zu betrachten, werden die Software-Benutzer-Programmierer ins Zentrum der Untersuchung gerückt. Im Zentrum der Analyse des Open-Source-Phänomens stehen dann die ‚Prosumer' (siehe Toffler 1980), d. h. Benutzer, welche die Software an ihre Bedürfnisse anpassen und weiterentwickeln. Wie von Hippel und von Krogh (2002) gezeigt haben, ist mit einem solchen Perspektivenwechsel ein wesentlicher Erkenntnisgewinn möglich.

Das heißt aber nicht, dass Prosumer weniger rational handeln als Softwarefirmen. Für beide Arten von Aktoren gilt, dass Software dann freigegeben wird, wenn der Nutzen einer Offenlegung des Quellcodes größer ist als die Kosten einer solchen Handlung. Nur präsentiert sich das Umfeld und damit auch die Rechnung für die Softwarefirmen anders als für die Prosumer. Softwarefirmen befinden sich in einem ausgeprägten Konkurrenzverhältnis zueinander. Für sie ist eine Freigabe des Quellcodes mit hohen Opportunitätskosten verbunden: Mit einer Offenlegung des Quellcodes geben sie ein Geschäftsgeheimnis preis und verspielen dadurch einen Wettbewerbsvorteil.

Für die Software-Benutzer-Programmierer sieht die Lage ganz anders aus. Das Verhältnis der Prosumer untereinander ist durch bloß geringe Rivalitätsbedingungen bestimmt. Zusätzlich gilt, dass dank dem Internet auch die direkten Kosten gering sind: die Verbreitung des Quellcodes verursacht nur minimale Kosten.

Doch der geringe Aufwand auf der Kostenseite kann das Verhalten der Software-Benutzer-Programmierer noch nicht erklären. Für rationale Aktoren wäre es immer noch sinnvoller, als Trittbrettfahrer am Open-Source-Phänomen teilzunehmen, d. h. ohne eigene Leistungen von den Arbeiten anderer zu profitieren, statt aktiv an der Erzeugung solcher Software mitzuwirken. Für einen Beitragsleister muss in irgendeiner Form ein selektiver Vorteil existieren, ein Nutzen, den nur jene Personen genießen können, die sich engagieren. Allerdings gilt in Situationen, wo die Kosten einer Freigabe gering sind (sog. „low cost"-Situationen), dass solche selektiven Vorteile nicht mehr groß zu sein brauchen, damit für die Benutzer-Programmierer ein Engagement für Open-Source-Software vorteilhaft erscheint.

Wenn es nun gelingt, solche selektiven Vorteile, von welchen nur die Prosumer, nicht aber die Trittbrettfahrer profitieren können, zu identifizieren und zu quantifizieren, so ist das eingangs beschriebene Paradox im Wesentlichen aufgelöst.

2. Motivationen

Welche selektiven Vorteile könnten also für Benutzer-Programmierer relevant sein und diese zu einem Engagement für Open-Source-Software motivieren?

In der wissenschaftlichen Literatur über Open Source finden sich etliche Analysen, die sich mit diesem Thema auseinandersetzen. Auf Grund dieser Untersuchungen konnte ein ganzes Bündel von Anreizen identifiziert werden. Die mittlerweile vorliegenden empirischen Studien über das Open-Source-Phänomen (Robles u.a.

2001, Jorgensen 2001, Hars und Ou 2001, Dempsey u.a. 2002, Ghosh u.a. 2002, Krishnamurthy 2002, Bonaccorsi und Rossi 2003, Hertel u.a. 2003, Healy und Schussman 2003, Lakhani und Wolf 2003) konnten die Existenz dieser unterschiedlichen Motivationen tatsächlich bestätigen (siehe Tabelle 1). In den folgenden Kapiteln werden diese Motivationen genauer erläutert.

2.1. Gebrauch

Die einfachste Begründung dafür, dass sich ein Softwareentwickler für ein Open-Source-Projekt engagiert, ist, dass er die von diesem Projekt erzeugte Software gebrauchen kann. Dieses Motiv entspringt dem Prosumer-Modell: Der Aktor hat ein Problem, welches mit der geeigneten Software gelöst werden kann, und erzeugt die entsprechende Software oder passt eine existierende für seine Bedürfnisse an. In der Studie von Lakhani und Wolf (2003) erhielt dieses Motiv die größte Zustimmung.

2.2. Reputation und Signalproduktion

Raymond hat in seinem letzten Essay „Homesteading the Noosphere“ (2000) aus seiner vielbeachteten „The Cathedral and the Bazaar“-Trilogie beschrieben, wie die Normen und Tabus der Open-Source-Bewegung in Bezug auf den Erwerb von Reputation wirken. Wie Raymond aufzeigt, sind „code forking“, d.h. das Aufsplitten der Code-Basis in inkompatible Versionen, und vor allem das Löschen der Namen der Beitragsleister aus den „credit files“ der Applikationen streng verpönt. Die Sensibilität der Open-Source-Community gegenüber diesen Normen macht Sinn im Hinblick auf den Aufbau von Reputation, denn ohne diese Normen wäre es viel schwieriger nachzuvollziehen, welcher Beitrag von welcher Person kommt, und damit wäre der Aufbau von Reputation stark erschwert.

Lerner und Tirole (2001) haben diesen Zusammenhang unter dem Aspekt der Signalproduktion analysiert. Gemäß dieser Argumentation ermöglicht es die Offenlegung des Quellcodes zusammen mit den spezifischen Normen der Open-Source-Community wie von Raymond dargestellt, dass die Beiträge der einzelnen Entwickler sehr gut verfolgt werden können. Dies hat zur Folge, dass der Status eines Entwicklers in einem Projekt ein genaues Abbild seiner Reputation ist und diese wiederum sehr transparent die Qualität und Quantität seiner Beiträge reflektiert. Vor diesem Hintergrund besteht nun die Möglichkeit, diese Reputation zu monetarisieren, z.B. durch ein interessantes Arbeitsangebot oder einen verbesserten Zugang zu Risikokapital. Der Arbeits- bzw. Risikokapitalmarkt ist primär an Talent und Leistung und nicht an Reputation interessiert. Das Talent einer Person ist aber für Personen, die nicht mit dem Fachgebiet vertraut sind, schwer einzuschätzen. Wenn allerdings die Reputation einer Person ein gültiger Indikator für deren Talent ist, kann Reputation im oben beschriebenen Sinn als Signal wirken: Aus dem Wissen über ein Open-Source-Projekt und der Position einer Person in diesem Projekt kann z.B. ein potentieller Arbeitgeber gültige Rückschlüsse auf das Talent der Person ziehen. Signalproduktion wirkt am stärksten, wenn die technische Herausforderung groß ist, wenn die relevante Öffentlichkeit (d.h. die Peergruppe) technisch erfahren ist, zwischen guten und herausragenden Leistungen unterscheiden sowie Leistung und Können

wertschätzen kann (Weber 2000, Franck und Jungwirth 2001). Diese Bedingungen sind im Falle von Open Source in hohem Maß gegeben.

2.3. Identifikation mit der Gruppe

Ist eine Person gut in eine Gruppe eingebunden und kann sie sich stark mit den Gruppenzielen identifizieren, so kann häufig ein großes Engagement dieser Person für die Gruppenziele beobachtet werden (Kollock und Smith 1999). Hertel u.a. (2002) untersuchten in einer empirischen Studie unter Linux-Entwicklern, ob dieses Phänomen auch im Open-Source-Bereich von Bedeutung ist. Sie haben die Entwickler über verschiedene Aspekte ihrer Tätigkeit befragt und diese Angaben mit den Werten über das Engagement korreliert. Mit statistischen Methoden gelang es ihnen tatsächlich nachzuweisen, dass ein signifikanter Anteil des Engagements der Entwickler durch ihre Identifikation mit ihrer Entwicklergemeinschaft erklärt werden kann.

Dieses Resultat ist von Lakhani und Wolf (2003) bestätigt worden. In ihrem Hacker-Survey untersuchten Lakhani und Wolf, wie sich die Identifikation mit der Projektgruppe auf das zeitliche Engagement auswirkt, und konnten dabei einen signifikant positiven Effekt feststellen.

2.4. Lernen

Der Einsatz für ein Open-Source-Projekt kann auch unter dem Aspekt des Lernens erklärt werden. Der Einsatz von neuster Technologie zur Bewältigung eines komplexen Problems ist für alle Beteiligten mit einem Gewinn an Erfahrung verbunden. Der Wunsch, seine Fähigkeiten als Softwareentwickler zu verbessern, taucht sowohl in der FOSS Studie von Ghosh u.a. (2002) wie auch im Hacker-Survey von Lakhani und Wolf (2003) und in der Untersuchung von Hars und Ou (2001) mit hohen Zustimmungsraten auf.

2.5. Altruismus

Ein Engagement für Open Source kann auch begründet werden mit einem Gefühl für das Wahre und Richtige, z.B. dass die Freiheit der Menschen mit quelloffener Software zusammenhängt und dieses Gut gefährdet ist durch kommerzielle und proprietäre Softwareanbieter. Dieser Standpunkt wird pointiert von Stallman und seiner Free Software Foundation (FSF) vertreten.

Um eine ähnliche Motivation handelt es sich, wenn ein Programmierer an einem Open-Source-Projekt beteiligt ist, weil er selbst viele Open-Source-Produkte verwendet und nun die Verpflichtung spürt, selbst etwas für die Open-Source-Bewegung zu tun.

Während das erste Motiv auf ein abstraktes Gefühl und eine ideologische Haltung verweist, referiert das zweite Motiv auf ein Gefühl des gerechten Gebens und Nehmens, einer allgemeinen Reziprozität im Verhältnis des Individuums zu einer Gruppe. Ökonomisch können die auf diese Weise motivierten Beiträge als Spenden interpretiert werden. Ob der Beitrag geleistet wird oder entfällt, hat für den Programmierer keine nachvollziehbaren Auswirkungen. Das Open-Source-Projekt, in

welchem sich der Spender engagiert, profitiert aber von den Beiträgen dieser Person.

Gemäß der Studie von Lakhani und Wolf (2003) bezeichneten ungefähr ein Drittel der befragten Open-Source-Programmierer diese Motive als relevant für ihr Engagement. Interessant ist, dass Lakhani und Wolf mit einer Cluster-Analyse zeigen konnten, dass das ideologisch fundierte Motiv, Software müsse offen sein, tatsächlich sehr nahe beim Motiv liegt, welches das Engagement aus dem Gefühl der Reziprozität heraus begründet.

2.6. Spaß

Seit es Computer gibt, üben diese Geräte, vor allem auf Männer, eine große Faszination aus. Die Schilderungen der Befriedigung und des Spaßes, den die Programmierer bei ihrer Tätigkeit empfinden, ziehen sich wie ein roter Faden durch die Lebensgeschichten von Computerexperten und Hackern. Brooks, der als Computerwissenschaftler und Softwarepionier bei IBM wirkte, beschreibt diesen Sachverhalt wie folgt: „Programming [then] is fun because it gratifies creative longings built deep within us and delights sensibilities we have in common with all men." (1995, S. 8). Linus Torvalds betitelt seinen Rückblick über die Entstehung von Linux schlicht mit „Just for FUN" (2001).

Doch die Tatsache, dass Programmieren Spaß macht, reicht noch nicht aus als Erklärung dafür, das Open-Source-Software programmiert wird. Auch Spaßsucher sind primär rational denkende und handelnde Personen. Wenn Programmieren Spaß macht, dann ist es immer noch besser, mit dieser Tätigkeit Geld zu verdienen, als das Resultat der Arbeit zu verschenken. Um die Bedeutung von Spaß wirklich zu verstehen, muss noch gezeigt werden, dass und wie sich Programmieren in einem Open-Source-Projekt unterscheidet von der Tätigkeit eines Softwareentwicklers in einem kommerziellen Unternehmen.

Solche Unterschiede können tatsächlich identifiziert werden:

- Der Projektmanager in einem kommerziellen Umfeld ist üblicherweise nicht diejenige Person, welche die Vision über die Applikation entwickelt hat und pflegt.
- Der Projekteigentümer in einem Open-Source-Projekt hat keine formale Autorität.
- Die Programmierer in einem Open-Source-Projekt können üblicherweise nicht über direkte monetäre Anreize zu einem Engagement oder zur Steigerung ihres Engagements bewegt werden.
- Ein Open-Source-Projekt hat üblicherweise keine Abgabetermine.

Solche Differenzen machen es nachvollziehbar, dass das Open-Source-Entwicklungsmodell mehr Spaß zulässt, als unter kommerziellen Produktionsbedingungen möglich ist.

Die Bedeutung von Spaß als Motivation für ein Engagement im Open-Source-Bereich wird von den bisherigen empirischen Studien weitgehend bestätigt. Beispielsweise ergaben die Fragen nach Kreativität und Flow in der Studie von Lakhani und Wolf (2003) außerordentlich hohe Zustimmungsraten von 61% respektive 73%.

2.7. Verträglichkeit der unterschiedlichen Motivationstypen

Osterloh u. a. (2002) unterteilen die möglichen Motivationen auf der ersten Stufe in intrinsische und extrinsische Motivationen (z.B. Signalproduktion, Gebrauch). Innerhalb der intrinsischen Motivationen unterscheiden sie wiederum in solche, die auf Freude basieren („enjoyment based", z.B. Spaß) und solche, die auf Verpflichtungen gegenüber der Gemeinschaft basieren („obligation"/„community based", z.B. Gruppenidentifikation).

Eine weitere Möglichkeit, die unterschiedlichen Motive zu kategorisieren, haben Franck und Jungwirth (2002b) vorgeschlagen. Sie unterscheiden zuerst zwischen Spendern und Rentensuchern. Als Spender werden diejenigen Beitragsleister bezeichnet, die sich aus ideellen oder altruistischen Gründen für Open Source engagieren. Rentensucher dagegen stellen sich die Frage, ob sich das Engagement für sie rentiert. Diese zweite Kategorie kann nochmals unterteilt werden in Investoren und Konsumenten. Für einen Entwickler, der sich im Sinne einer Investition in einem Open-Source-Projekt engagiert, zahlt sich das Engagement erst im Verlauf der Zeit z.B. als gestiegene Reputation oder als Folge der größeren Fähigkeiten aus. Ein Entwickler, der sich aus Freude am Programmieren beteiligt oder die Software zur Lösung eines Problems braucht, zieht dagegen einen unmittelbaren Nutzen aus seiner Tätigkeit.

Wesentlich ist, dass sich die unterschiedlichen Motive nicht gegenseitig ausschließen, sondern, im Gegenteil, ergänzen. Dieser Umstand liegt im Wesen der Open-Source-Lizenz begründet. Eine Open-Source-konforme Lizenz unterstellt die Software einer „Non Distribution Constraint" (siehe dazu Hansmann 1980). Damit ist sichergestellt, dass die Beiträge der Spender nicht entgegen deren Intention proprietär vereinnahmt werden können. Aber auch für Rentensucher, die im Hinblick auf ihre Reputation investieren, ist die Open-Source-Lizenzbedingung von Bedeutung. Erst die Garantie, dass der Quellcode offen bleibt, ermöglicht den transparenten Nachweis der Leistung und fundiert damit die Reputation der Beitragsleistenden.

Weiter ist es für Rentensucher, die im Hinblick auf die Signalproduktion in ihre Reputation investieren, rational, in reife Projekte zu investieren, welche schon einen ersten Markttest bestanden haben oder kurz davor sind, einen gewissen Marktanteil zu erobern. Mit neuen Open-Source-Projekten dagegen können noch keine relevanten Signale produziert werden. Durch Reputation extrinsisch motivierte Entwickler brauchen also Spender oder „Fun seeker", damit ein Open-Source-Projekt überhaupt entsteht. Umgekehrt ist die Teilnahme von profilsuchenden Investoren auch für Spender vorteilhaft. Wer Signale produzieren will, ist eher bereit, Kundenwünsche aufzunehmen und diese in das Projekt zu integrieren. Das Eingehen auf Kundenbedürfnisse ist aber für den langfristigen Erfolg eines Softwareprodukts unabdingbar.

Diesen Sachverhalt, dass in der Open-Source-Bewegung Personen mit ganz unterschiedlichen Motivationen beteiligt sind und miteinander kooperieren, konnten Lakhani und Wolf (2003) in ihrer empirischen Studie bestätigen.

3. Das Forschungsprojekt „Fun and Software Development“

Spaß als Motivation für ein Engagement in ein Open-Source-Projekt ist der Gegenstand des Forschungsprojekts „Fun and Software Development“ (FASD). Ziel dieses Forschungsprojekts ist es, eine möglichst genaue quantitative Abschätzung über die Bedeutung dieses Motivationsfaktors zu erzielen.

3.1. FASD-Modell

Im FASD-Forschungsprojekt wird ein Modell benutzt, welches einen Zusammenhang zwischen Spaß und Engagement herstellt. Dieses Modell orientiert sich an der Theorie kompensierender Lohndifferenziale (Lorenz und Wagner 1988, Backes-Gellner u. a. 2001). Diese Theorie erklärt nachweisbare Lohnunterschiede durch das Vorhandensein von nicht-monetären Faktoren. So honorieren z.B. Angestellte größere Flexibilität bei der Arbeitszeitgestaltung durch Zurückhaltung bei Lohnforderungen oder wollen ein größeres Unfallrisiko am Arbeitsplatz durch höheren Lohn entschädigt haben. Angewendet auf die Aufgabe, das Phänomen Open Source zu erklären, besagt das Modell, dass Programmierer für weniger Lohn arbeiten, je mehr Spaß die Tätigkeit bereitet. Im Idealfall der Open-Source-Situation macht das Entwickeln von Software so viel Spaß, dass eine monetäre Entschädigung überhaupt keine Rolle mehr spielt. Wenn im typischen Open-Source-Fall ohne Lohn programmiert wird, so handelt es sich dabei um eine Freizeitbeschäftigung. Eine Tätigkeit in der Freizeit ist aber nur möglich, wenn überhaupt Freizeit verfügbar ist. Wir müssen unser Modell also durch den Faktor „Freizeit“ ergänzen.

Auf Grund dieser Überlegungen besagt das Modell, welches mit der FASD-Studie überprüft werden soll, Folgendes: Je mehr Spaß das Programmieren macht und je mehr Freizeit der Softwareentwickler hat, desto größer ist sein Einsatz beim Entwickeln von Open-Source-Software.

3.2. Flow-Konzept

Zur Ermittlung des Motivationsfaktors „Spaß“ wird für die FASD-Studie das von Csikszentmihalyi (1975) entwickelte Flow-Konzept verwendet. Flow erscheint aus zwei Gründen besonders tauglich, Spaß beim Programmieren zu messen. Einerseits ist das Empfinden von Flow als psychologisches Konstrukt schon relativ gut untersucht worden. Es existieren eine ganze Reihe von empirischen Studien zu Flow und ein entsprechendes Wissen, wie dieses Konstrukt gemessen werden kann. Andererseits scheint speziell die Tätigkeit des Softwareentwickelns gut geeignet, Flow-Empfindungen hervorzurufen. Ein großer Teil der Flow-Studien haben gerade Tätigkeiten mit und an Computern verwendet, um Flow zu erforschen.

Dieser Zusammenhang erstaunt nicht, wenn man die Komponenten betrachtet, aus denen Flow besteht (zitiert aus Rheinberg 1997, S. 143):

- Handlungsanforderungen und Rückmeldungen werden als klar und interpretationsfrei erlebt, sodass man jederzeit und ohne nachzudenken weiß, was jetzt als richtig zu tun ist.
- Man fühlt sich optimal beansprucht und hat trotz hoher Anforderungen das sichere Gefühl, das Geschehen noch unter Kontrolle zu haben.

- Der Handlungsablauf wird als glatt erlebt. Ein Schritt geht flüssig in den nächsten über, als liefe das Geschehen gleitend wie aus einer inneren Logik. (Aus dieser Komponente rührt wohl die Bezeichnung „Flow".)
- Man muss sich nicht willentlich konzentrieren, vielmehr kommt die Konzentration wie von selbst, ganz so wie die Atmung. Es kommt zur Ausblendung aller Kognitionen, die nicht unmittelbar auf die jetzige Ausführungsregulation gerichtet sind.
- Das Zeiterleben ist stark beeinträchtigt; man vergisst die Zeit und weiß nicht, wie lange man schon dabei ist. Stunden vergehen wie Minuten.
- Man erlebt sich selbst nicht mehr abgehoben von der Tätigkeit, man geht vielmehr gänzlich in der eigenen Aktivität auf (sog. „Verschmelzung" von Selbst und Tätigkeit). Es kommt zum Verlust von Reflexivität und Selbstbewusstheit.

In der Idealsituation für einen Softwareentwickler, wenn sich der Programmierer ungestört seiner selbstgewählten Aufgabe widmen kann, wenn er vor einem auf seine Bedürfnisse zugeschnittenen und konfigurierten Computer sitzt, ein Stück seines Projekts implementiert, dieses testet, die Ergebnisse analysiert und auf Grund der Rückmeldungen sein Werk verbessert, seinen neuen Code in die bestehende Software integriert und den nächsten Teil in Angriff nimmt, wenn sich unter seinen Händen ein raffiniertes Stück Software bildet und dieses Schritt für Schritt wächst, an Eleganz und Leistungsfähigkeit gewinnt, in einer solchen Situation sind die Voraussetzungen für das Erleben von Flow optimal.

Eine in Flow-Studien oft verwendete Methode besteht darin, den Probanden eine Reihe von Frage-Items vorzulegen (20 bis 40 Stück), die verschiedene Gefühlsempfindungen beschreiben. Die Probanden können dann auf einer Skala wählen, wie gut die Beschreibungen mit ihren Empfindungen übereinstimmen. Mit Hilfe von Faktoranalysen können diese Antworten in der Folge statistisch ausgewertet werden. Solche Analysen zeigen übereinstimmend, dass Flow als Konstrukt im Wesentlichen aus den Komponenten „Flow-Erleben", „Eindeutigkeit der Aufgabe" und „Erlebte Leichtigkeit" besteht (z.B. Remy 2000).

3.3. Online-Umfrage

Die in der FASD-Studie gewählte Methode orientiert sich an der oben beschriebenen Vorgehensweise. Die Online-Umfrage des FASD-Projekts besteht aus rund 30 Items, welche sich auf die Bestimmung von Flow beziehen. In diesem Teil werden die Probanden gefragt, wie häufig eine Empfindung wie beispielsweise „Alles scheint wie von selbst zu laufen" oder „Meine Aufmerksamkeit ist völlig auf die Handlung gelenkt" zutrifft, wenn sie Software entwickeln. Die Teilnehmer beantworten diese Fragen, indem sie einen Wert auf einer sechspunktigen Likertskala zwischen „nie" und „immer" wählen.

In den weiteren Teilen des Fragebogens werden die Probanden gefragt, wie sie ihr zukünftiges Engagement für Open-Source-Projekte einschätzen und welchen Einsatz sie bisher geleistet haben, wie sie die Arbeit in und Organisation von Open-Source-Projekten empfinden und welches ihr ursprünglicher Anlass war, sich in die-

sem Bereich zu betätigen. Im letzten Teil werden demographische Daten sowie Angaben über die Freizeitbeschäftigung der Probanden erfragt.

Das Besondere an der FASD-Studie ist, dass gleichzeitig Open-Source-Programmierer wie auch Entwickler an kommerziellen Projekten befragt werden. Diese Versuchsanordnung gestattet es in einem ersten Schritt zu verifizieren, ob die Programmierer tatsächlich mehr Spaß haben, mehr Flow erleben, wenn sie für Open Source entwickeln, als wenn sie diese Tätigkeit unter kommerziellen Bedingungen ausüben. Weiter ist es durch statistische Methoden möglich, das Modell zu überprüfen und auf diese Weise den Beitrag von Spaß am Engagement zu quantifizieren. Konkret werden die Angaben über das Flow-Erleben mit den Daten bezüglich Einsatzbereitschaft gemäß dem Modell korreliert, und daraus wird berechnet, welchen Anteil des Engagements durch das Modell erklärt werden kann. In einem letzten Schritt soll mit der Untersuchung gezeigt werden, dass die identifizierten Unterschiede bezüglich Projektmanagement, formaler Autorität, Anreizsystem und Abgabeterminen tatsächlich die ausschlaggebenden Faktoren sind für unterschiedliches Flow-Empfinden.

4. Weiterführende Fragen

Auf Grund der vorliegenden Forschungsresultate und der verschiedenen laufenden und geplanten Studien im Bereich Open Source bin ich der Ansicht, dass in absehbarer Zeit genügend Informationen vorhanden sein dürften, um die Motivation von Open-Source-Entwicklern hinreichend zu verstehen. Damit dürfte die Frage nach den Motiven an Interesse verlieren und sich der Forschungsschwerpunkt auf andere Phänomene im Open-Source-Bereich verlagern. Eine interessante und noch wenig erforschte Frage ist beispielsweise, unter welchen Umständen ein Open-Source-Projekt erfolgreich ist.

Das Engagement von Softwareprogrammierer ist keineswegs eine hinreichende Bedingung für einen Projekterfolg. Die empirischen Studien von Krishnamurthy (2002) und Healy und Schussman (2003) belegen, dass ein erheblicher Anteil von Open-Source-Projekten kaum das Anfangsstadium überwindet. Der Großteil der Projekte hat auch nach erheblicher Zeit noch keine Entwicklergemeinschaft anziehen können, statt dessen dämmern die Projekte als Ein-Personen-Vorhaben dahin und sind weit von einem nützlichen Einsatz entfernt.

Die Fragen nach den Erfolgsbedingungen von Open-Source-Projekten hängt möglicherweise eng mit der Frage zusammen, mit welchen Geschäftsmodellen unter welchen Bedingungen mit Open-Source-Software Geld verdient werden kann. Hecker (2000) hat eine Reihe von Geschäftsmodellen beschrieben, die nachvollziehbar aufzeigen, dass auch mit Software, deren Quellcode freigegeben ist, ein Profit erzielt werden kann. Allerdings fehlen noch weitgehend vertiefte Untersuchungen, ob die identifizierten Geschäftsmodelle auch wirklich funktionieren (siehe etwa Wichmann 2002). Die Untersuchung von Bonaccorsi und Rossi (2003) zeigt aber deutlich, dass sich die Motive, die eine Firma veranlassen, sich im Open-Source-Bereich zu engagieren, klar von denjenigen der einzelnen Hacker unterscheiden.

Sollte sich nun, z.B. mit Hilfe der FASD-Studie die Hypothese erhärten, dass Open-Source-Projekte im Wesentlichen durch Spass motiviert sind, so könnten sich die Geschäftsmodelle als das Verbindungsglied erweisen, welche den Erfolg von Open-Source-Projekten erklären. Softwareentwickler, die Spaß suchen und als Resultat dieses Verlangens Programme herstellen, befriedigen in erster Linie ihre eigenen Bedürfnisse. Erfolgreiche Software, ob unter einer Open-Source-Lizenz oder proprietär, zeichnet sich aber dadurch aus, dass sie die Bedürfnisse einer möglichst großen Benutzerschaft erfüllt. Erst der Wunsch, mit der Software auch etwas zu verdienen, zwingt die Softwareentwickler, auch noch andere als die eigenen Bedürfnisse wahrzunehmen. Und erst eine zufriedene Kundschaft führt zu einer wahrnehmbaren Marktdurchdringung des Open-Source-Produkts und damit zu dessen Erfolg.

5. Zusammenfassung

In diesem Beitrag habe ich gezeigt, dass sich das Paradox „Warum gibt eine Person ihre Software frei, wenn mit dieser Software viel Geld verdient werden könnte?“ auflöst, wenn als Aktoren nicht Profit maximierende Softwarefirmen, sondern die einzelnen Open-Source-Hacker untersucht werden. Solche Softwareentwickler sehen sich einer Low-Cost-Situation gegenüber. In dieser Situation ist es mit geringen Anreizen möglich, die Aktoren zu einem Beitrag für ein öffentliches Gut zu bewegen.

Ein Softwareentwickler kann aus einem ganzen Bündel an Motivationen auswählen, die auf unterschiedliche Weise dahin wirken, dass es für einen Programmierer vorteilhaft ist, sich in einem Open-Source-Projekt zu engagieren. In diesem Beitrag habe ich als Motive Gebrauch, Signalproduktion, Gruppenidentifikation, Lernen, Spenden und Spaß vorgestellt. Die vorliegenden empirischen Studien über das Open-Source-Phänomen bestätigen die Existenz dieser Motive. Es besteht allerdings noch keine Klarheit, wie relevant die einzelnen Motivationen zur Erklärung des Engagements der Open-Source-Hacker sind.

Diese Lücke soll mit dem Projekt FASD, zumindest teilweise, geschlossen werden. Ziel dieses Forschungsprojekts ist es, die Bedeutung der Motivation „Spaß“ quantitativ zu bestimmen. Spaß wird im FASD-Projekt mit Hilfe des Flow-Konzepts von Csikszentmihalyi untersucht. Mit Hilfe eines Online-Fragebogens soll verifiziert werden, dass Programmieren in einem Open-Source-Projekt mehr Spaß macht als unter den Bedingungen in einer kommerziellen Firma, und dass die für Open-Source-Projekte wesentlichen Eigenheiten bestimmend sind für diesen Unterschied. Weiter soll mit diesen Daten untersucht werden, welcher Anteil des Engagements mit dem auf Spaß basierenden Modell erklärt werden kann.

Kann durch die FASD-Studie gezeigt werden, dass Spaß ein zentraler Motivationsfaktor ist, so kann Open-Source-Software interpretiert werden als mehr oder weniger intendiertes Nebenprodukt einer Tätigkeit, die eminent Spaß macht. Damit stellt sich aber die Frage, warum und unter welchen Umständen Open-Source-Software erfolgreich ist. Zur Klärung dieser Frage müssen die möglichen Geschäftsmodelle für Open-Source-Software untersucht werden. Damit geraten die Firmen, die

Open-Source-Software in irgendeiner Form unterstützen, wieder ins Blickfeld der wissenschaftlichen Untersuchung.

Während also zur Erforschung der Frage, warum Open-Source-Software entsteht, die Benutzer-Programmierer untersucht werden, sind es bei der Frage, warum Open-Source-Projekte erfolgreich sind, die Firmen und ihre Geschäftsmodelle.

Anhang

Tabelle 1: Empirische Untersuchungen zu Open Source

Studie und Thema	Methode	Sample-Größe
Robles u. a. (2001); Merkmale von Open-Source-Entwicklern	Online-Fragebogen	5478
Ergebnisse: 80% der OS-Entwickler sind im IT-Bereich tätig, 33% im universitären Bereich. 21% der OS-Entwickler verdienen Geld mit dieser Tätigkeit. 54% der OS-Entwickler haben von Engagement im Open Source-Bereich beruflich nicht profitiert, wobei 26% hoffen, dies werde in Zukunft geschehen. 46% habe profitiert, wovon 15% ihre aktuelle Position diesem Engagement verdanken und 2% eine Lohnsteigerung.		
Jorgensen (2001); Wirkung des Open-Source-Entwicklungsmodells auf Motivation	Online-Fragebogen und Einzelinterviews	72
Ergebnisse: 43% der Befragten wurden für ihre Arbeit am FreeBSD-Projekt ganz oder teilweise bezahlt. 86% der Befragten geben an, dass ihre Beiträge einen Review erhalten haben. Gemäß 57% der Befragten haben die Reviews der eigenen Beiträge dazu geführt, dass ihre Fähigkeiten als Entwickler deutlich verbessert wurden.		
Hars und Ou (2001); Motivation von Open-Source-Entwicklern	Online-Fragebogen	81
Ergebnisse: Als Motivation bewerteten 80% der Befragten Selbstbestimmung als hoch bis sehr hoch, für 88% war die Bildung von Humankapital wichtig. Selbstbestimmung ist für 93% der OS-Entwickler aus dem IT-Bereich, die in ihrer Freizeit OS entwickeln, überdurchschnittlich motivierend, für bezahlte OS-Programmierer mit 61,5% nur unterdurchschnittlich. Der Lerneffekt ist für Studierende und Hobbyprogrammierer überdurchschnittlich wichtig (97%).		
Dempsey u. a. (2002); Mermale von Linux-Entwicklern	Analyse von Linux Software Maps (LSM)	4.633
Ergebnisse: Anzahl Beiträge pro Entwickler: Von den insgesamt 2.429 registrierten Entwicklern leisteten 91% 1 bis 2 Beiträge, 2,2% mehr als 5 Beiträge. Nur von 13 Entwickler stammten mehr als 10 Beiträge.		

Studie und Thema	Methode	Sample-Größe
Ghosh u. a. (2002); Merkmale von Open-Source-Entwicklern	Online-Fragebogen	2.784
Ergebnisse: Alter der OS-Entwickler: Mittelwert: 27 Jahre, 25% älter als 30 Jahre. Nationalität: 71% Europa (16,5 Frankreich, 12,4 Deutschland), 13% Nordamerika (10,3% USA). Zeitliches Engagement pro Woche: 23% < 2 h, 70% < 10 h, 14 % > 10 h und < 20 h, 9% > 20 h und < 40 h, 7% > 40 h. Motive für Engagement in OS: 80% neue Fähigkeiten erwerben, 50% Wissen teilen, 33% Kooperation in OS, 33% Verbesserung von bestehender OSS, 30% OSS als öffentliches Gut.		
Krishnamurthy (2002); Merkmale von Open-Source-Projekten	Analyse der 100 aktivsten SourceForge-Projekte	100
Ergebnisse: Die 100 erfolgreichsten SourceForge-Projekte haben im Durchschnitt eine Gruppengröße von 6,6 Entwicklern (Median: 1 Entwickler). Je mehr Entwickler ein Projekt hat, desto mehr wird es beachtet (Korrelationskoeff. 0,56) und desto häufiger erfolgen Downloads (0,27).		
Bonaccorsi und Rossi (2003); Merkmale von Firmen im Open-Source-Bereich	Fragebogen	146
Ergebnisse: Die befragten Firmen bewerteten auf einer fünfpunktigen Likertskala (Wertung von 1 bis 5) die ökonomischen Motive am höchsten (3,56), vor den technologischen (3,51) und den sozialen (3,39).		
Hertel u. a. (2003); Motivation unter Linux-Kernel-Entwicklern	Online-Fragebogen	141
Ergebnisse: Verschiedene Motivationsfaktoren (Identifikation mit dem Projekt, Spaß, Problemlösung etc.) wurden mit einer fünfpunktigen Likertskala (Wertung von 1 bis 5) bewertet und die Antworten mit dem Engagement für das Projekt korreliert. Spaß wurde mit 4,6 am höchsten bewertet. In der Varianzanalyse erhielt das Motiv „Identifikation mit der Projektgruppe“ mit 22,9% den höchsten Erklärungsgehalt. Das Ergebnis ist statistisch signifikant.		
Healy und Schussman (2003); Merkmale von Open-Source-Projekten	Analyse von SourceForge-Projekten	46.356
Ergebnisse: Projektgröße: Mittelwert: 1,7 Entwickler, Median: 1 Entwickler, 95-Perz.: 5 Entwickler. Anzahl Downloads: Mittelwert: 2289, Median: 0, 90-Perz.: 872.		
Lakhani und Wolf (2003); Motivation von Open-Source-Entwicklern	Online-Fragebogen	648

Studie und Thema	Methode	Sample-Größe
Ergebnisse: Die befragten Entwickler verwenden im Durchschnitt 14 Stunden pro Woche für die Entwicklung von OSS. 61% der Befragten erlebten bei ihrer OS-Tätigkeit die kreativsten Momente in ihrem Leben. 73% erlebten häufig oder immer Flow-Zustände bei ihrer OS-Tätigkeit. Motivation für das Engagement: Gebrauch 59%, Spaß 45%, Lernen 41%, Altruismus 33%. Die verschiedenen Motivationstypen wurden mit dem zeitlichen Engagement korreliert und der Effekt der einzelnen Beiträge wurde mittels Regressionsanalyse berechnet: Kreativität: 1,6, Bezahlung: 0,9, Freude an Teamarbeit: 0,8, Reputation: 0,6, Arbeit in anderen FOSS-Projekten: -1,6, IT-Ausbildung: -0,6 ($r^2 = 0{,}18$)		

Literaturverzeichnis

Backes-Gellner, U. / Lazear, E. P. / Wolff, B. (2001): *Personalökonomik. Fortgeschrittene Anwendungen für das Management*, Stuttgart

Bonaccorsi, A., C. Rossi (2003): *Altruistic individuals, selfish firms? The structure of motivation in Open Source software*,
online
http://opensource.mit.edu/papers/bnaccorsirossimotivationshort.pdf

Brooks, Frederick P. (1995): *The Mythical Man-Month: Essays on Software Engineering*, Reading, Massachusetts

Csikszentmihalyi, M. (1975): *Beyond boredom and anxiety*, San Francisco

Csikszentmihalyi, M. (1990): *Flow. The Psychology of Optimal Experience*, New York

Dempsey, B. / Weiss, D. / Jones, P. / Greenberg, J. (2002): *Who is an Open Source Software Developer?*, Communications of the ACM 45:2, p. 67–73.

Franck, E. / Jungwirth, C. (2001): *Open versus Closed Source. Eine organisationsökonomische Betrachtung zum Wettbewerb der Betriebssysteme Windows und Linux*;
online http://www.unizh.ch/ifbf/webFuehrung/Dokumente/Working Paper/FranckJungwirthOpenversusClosedSourceWP4.pdf

Franck, E., / Jungwirth, C. (2002a): *Reconciling investors and donators. The governance structure of open source*,
online http://www.unizh.ch/ifbf/webFuehrung/Dokumente/Working Paper/FranckJungwirthInvestorsAndDonatorsWP8.pdf

Franck, E. / Jungwirth, C. (2002b): *Die Governance von Open-Source-Projekten*,
online http://www.unizh.ch/ifbf/webFuehrung/Dokumente/Working Paper/FranckJungwirthGovernanceofOpenSourceWP9.pdf

Franke, N. / von Hippel, E. (2002): *Satisfying Heterogenous User Needs via Innovation Toolkits: The Case of Apache Security Software*,
online http://opensource.mit.edu/papers/frankevonhippel

Ghosh, R.A. / Glott, R. / Krieger, B. / Robles, G. (2002): *FLOSS: Free/Libre and Open Source Software: Survey and Study*,
online http://www.infonomics.nl/FLOSS/report/

Hansmann, H. B. (1980): *The role of nonprofit enterprise*, Yale Law Journal 89, 835–901

Hars, A. / Ou, S. (2001): *Working for Free? – Motivations of Participating in Open Source Projects*, 34th Annual Hawaii International Conference on System Sciences
Healy, K. / Schussman, A. (2003): *The Ecology of Open Source Software Development*, online http://opensource.mit.edu/papers/healyschussman.pdf
Hecker, F. (2000): *Setting Up Shop: The Business of Open-Source Software*, online http://www.hecker.org/writings/setting-up-shop.html
Hertel, G. / Niedner, S. / Herrmann, S. (2003): *Motivation of Software Developers in Open Source Projects*, Research Policy 32:7, S. 1159–1177
Jorgensen, N. (2001): *Putting it All in the Trunk*, Information Systems Journal 11:4
Krishnamurthy, S (2002): *Cave or Community? An Empirical Examination of 100 Mature Open Source Projects*, online http://www.firstmonday.org/issues/issue7_6/krishnamurthy/index.html
Lakhani, K. / Wolf, R. (2003): *Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects*, online http://opensource.mit.edu/papers/lakhaniwolf.pdf
Lerner, J. / Tirole, J. (2001): *The Simple Economics of Open Source*, online http://opensource.mit.edu/papers/
Lorenz, W. / Wagner, J. (1988): *Kompensierende Lohndifferentiale*, WiSt-Wirtschaftswissenschaftliches Studium 17:10, S. 515–518
Osterloh, M. / Rota, S. / Kuster, B. (2002): *Open Source Software Production: Climbing on the Shoulders of Giants*, online http://opensource.mit.edu/papers/osterlohrotakuster.pdf
Raymond, E. S. (2000): *Homesteading the Noosphere*, online http://www.catb.org/~esr/writings/homesteading/
Remy, K. (2002): *Entwicklung eines Fragebogens zum Flow-Erleben*, Bielefeld: Diplomarbeit (Fakultät für Psychologie und Sportwissenschaft)
Rheinberg, F. (1997): *Motivation*, Stuttgart
Robles, G. / Scheider, H. / Tretkowski, I. / Weber, N. (2001): *Who is doing it? A research on Libre Software developers*, online http://widi.berlios.de/paper/study.html
Toffler, A. (1981): *The Third Wave*, New York
Torvalds, L. / Diamond, D. (2001): *Just for FUN. The Story of an Accidental Revolutionary*, New York
Hippel, E. von / Krogh, G. von (2002): *Exploring the Open Source Software Phenomenon: Issues for Orgnization Science*, online http://opensource.mit.edu/papers/removehippelkrogh.pdf
Weber, Steven (2000): *The Political Economy of Open Source Software*, online http://brie.berkeley.edu/~briewww/pubs/wp/wp140.pdf
Wichmann, T. (2002): *FLOSS Final Report – Part 2: Firms' Open Source Strategy: Motivations and Policy Implications*, online http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf