Open-Source-Software – Vortrag

Definition von Open-Source

Der englische Ausdruck Open Source steht einerseits für "quelloffen" (in dem Sinne, dass der Quelltext eines Programms frei erhältlich ist), andererseits für 'offene Quelle' (in dem Sinne, dass ein Werk frei zur Verfügung steht).
Software gilt als Open Source, wenn sie folgende Kriterien erfüllt:

1. Freie Weitergabe
Die Lizenz darf niemanden darin hindern die Software zu verkaufen oder sie mit anderer Software zusammen in einer Software-Distribution weiterzugeben. Die Lizenz darf keine Lizenzgebühr verlangen.
2. Quellcode
Die Software muss im Quellcode für alle Nutzer verfügbar sein.
3. Abgeleitete Arbeiten
Die Lizenz muss von der Basissoftware abgeleitete Arbeiten und deren Distribution unter derselben Lizenz wie die Basissoftware erlauben.
4. Integrität des Autoren-Quellcodes
Die Lizenz muss explizit das Verteilen von Software erlauben die auf einer modifizierten Version des Originalquellcodes beruhen. Die Lizenz kann verlangen das solche Änderungen zu einem neuen Namen oder eine neue Versionsnummer der Software führen und solche Anderungen dokumentiert werden.
5. Keine Diskriminierungen von Personen oder Gruppen
Die Lizenz darf nicht einzelnen Personen oder Gruppen die Nutzung der Software verweigern.
6. Keine Nutzungseinschränkung
Die Lizenz darf den Verwendungszweck der Software nicht einschränken.
7. Lizenzerteilung
Die Lizenz muss für alle zutreffen, welche die Software erhalten, ohne z.B. eine Registrierung oder eine andere Lizenz erwerben zu müssen.
8. Produktneutral
Die Lizenz muss produktneutral gestaltet sein und darf sich z.B. nicht auf eine bestimmte Distribution beziehen.
9. Die Lizenz darf andere Software nicht einschränken
Sie darf zum Beispiel nicht verlangen das sie nur mit Open Source Software verbreitet werden darf.
10. Die Lizenz muss technologieneutral sein
Sie darf z.B. nicht verlangen das die Distribution nur via Web/CD/DVD verteilt werden darf.

Grundlagen und Hintergründe

Ab Mitte der 60ziger Jahre waren Quellcodes für Programmierer in der ganzen Welt zugänglich. Die Hersteller verdienten ausschließlich an der Hardware.
Einige Programmierer stellt dann fest, dass sich mit der entwickelten Software erhebliche Einkünfte erzielen ließen. Es wurden schließlich für die Software Lizenzverträge eingeführt.
Seither waren die Computer-Anwender bei Programmfehlern oder Sonderwünschen auf das entgegenkommen der Software-Firmen angewiesen.
1969 entwickelten Ken Thompson und Dennis Ritchie von den Bell Laboratories für die Telefongesellschaft AT&T in den USA die leistungsstarke Software UNIX. AT&T hatte Rechner verschiedener Hersteller gekauft für deren Betriebssysteme es keine einheitlichen

Standards gab. UNIX sollte die Verbindung zwischen den Computer vereinfachen. Heute wird UNIX als Ursprung der Open-Source-Software. Die Begriffe wie „Free Software“ und „Open Source“ entstanden aber erst später.
Nach der Erfindung der Programmiersprache C konnte die UNIX-Software auf verschiedenen Hardwareplattformen eingesetzt werden. Der große Durchbruch von UNIX war aber nicht die Plattformunabhängigkeit und der modulare Aufbau, sondern die frühe Integration von TCP/IP (Protokoll, auf dem das Internet basiert). Dadurch konnten sich Universitäten und Forschungsinstitute sich an dem UNIX-Projekt beteiligten.
1984 beschloss Richard Stallman vom MIT (Massachusetts Institute of Technology) ein freies Programmpaket namens GNU entwickeln. In Verbindung mit einem Betriebssystem-Kern (z. B. Linux) ergänzen sich die GNU-Tools zu einem kompletten Betriebssystem, umfassende Anwendungssoftware und eine vollständige Entwicklungsumgebung. Nach Ansicht von Stallmann sollte der Quellcode vervielfältig, verändert und weitergeben werden können. In seinen Augen bedeutete „free“, dass nicht nur der Quellcode frei sein muss, sodern auch die Modifikation (nicht im Sinne von Open-Source). Mit dieser Überzeugung gründeten Stallman und seine Mitstreiter die FSF (Free Software Foundation). Zusätzlich schuff er die GNU-GPL (GNU-General Public License), die die Freiheit der Software schützt. Die Idee, die hinter GPL steckt, ist das Copyleft; es erlaubt die uneingeschränkte Verteilung und Verwendung der unter seinem Schutz stehenden Programmes

- den Zugang zum Quellcode,
- die Freiheit, die Software zu kopieren und weiterzugeben,
- die Freiheit, das Programm zu ändern, und
- die Freiheit, das Programm unter denselben Bedingungen zu verbreiten.

Gleichzeitig stellt es klar, dass alle Software, die aus der Veränderung eines der ursprünglichen Programme entsteht, selbst wiederum unter das Copyleft fallen muss. Mit Hilfe dieser Klausel wollte Stallman die freie Software vor Besitzansprüchen und Patentierungsbestreben schützen.
Leider klang der Begriff „Free Software“ für viele verdächtig nach Freibier. Dadurch traten sie der Nutzung dieser Software mit Skepsis gegenüber. Mit dem idealistischen Ansatz von Stallman bleib die quelloffene Software von Wirtschaft und Unternehmen weitgehend ungenutzt. Weiter kam es zu dem Missverständnis, „free“ stünde für kostenlos und mit quelloffener Software ließe sich in keiner Weise Gewinn erwirtschaften und sei weniger vertrauenswürdig.
Der Software-Experte Eric S. Raymond schlug 1998 letztendlich vor die Software mit offenen Quellcode als ‚Open-Source-Software’ zu bezeichnen. Ende der neunziger gründete Raymond und Bruce Perens eine zweite Organisation, die OSI (Open Source Initiative).
Die Gründung dieser Initiative basierte auf das Ereignis, dass die Firma Netscape den Quellcode seines Webbrowsers zur offenen Programmierung freigegeben hatte. Damit war zum ersten Mal die offene Entwicklung eines Programms möglich, das auch bei Privatanwendern bekannt und weit verbreitet war
Zum Beginn der 90ziger Jahre waren allen wichtigen Bestandteile von GNU fertiggestellt. Es fehlte nur noch der „Kernel“, der stabile Betriebssystemkern. Unabhängig zu Stallman´s Projekt entwickelte der finnische Student Linus Torvalds von der Universität Helsinki 1991 einen freien UNIX-Kernel (Linus’ UNIX = Linux). Auf der Suche nach einer Möglichkeit den Kernel auszuprobieren, stieß man auf GNU. Das war die Geburtsstunde des Betriebssystems GNU/Linux.
Linux übernahm die Daten- und Speicherverwaltung sowie einige Low-Level Funktionen. Die großen Teile des Betriebssystems wie die grafische Oberfläche, Teile der Netzwerksoftware und Entwickler-Tools entstammen dem GNU-Projekt.
Weitere Projekte zur grafischen Benutzeroberfläche für alle UNIX-kompatibeln Systeme wie KDE (K Desktop Environment) und GNOME (GNU’s Network Object Model Environment) wurden ist Leben gerufen.

Grundideen, die die Open Source Bewegung maßgeblich beeinflusst haben

UNIX als Ausgangssystem und das Internet, das die Zusammenarbeit nach dem Open Source Prinzip ermöglicht, haben die Open Source Bewegung entscheiden geprägt.
Dann auch der Qualitätsanspruch, der an die jeweilige Software gestellt wird. Aber auch die Programmierung von Software, die für ein kommerziell orientiertes Unternehmen nie hergestellt werden würde, da sich mit dieser Software kein oder nur wenig Einkünfte erzielen ließen.

Ziele von Open-Source-Software

Stallman verfolgte das Ziel die offene Zusammenarbeit der Software-Entwickler erneut zu ermöglichen.
Ziel der OSI ist es

- Open-Source-Software der Wirtschaft näher zu bringen
- OS-Bewegung durch Zusammenarbeit stärken
- Geschäftsmodelle finden, mit denen mit OSS gewinne erwirtschaften lassen.

Hier wird deutlich, warum der Begriff „Open-Source“ an Stelle von „Free Software“ eingeführt wurde.

Stärken und Vorteile von Open-Source-Software

- **kostenloses Download** der Software aus dem Internet
- **weniger Virenanfällig**; Tatschache: mit Open-Source-Betriebssystemen ausgestatteten Rechner wurden von den Viren z.B. „Melissa“ oder „I Love You“ nicht beeinflusst
(Anmerkung: Viele Auge sehen zwar schnell einen Fehler [Fehlerbehebung], aber viele Augen können auch schnell ein Bufferoverflow in der Quellcode eintragen.)
- **Verfügbarkeit des Quellcodes und das Recht, ihn ändern zu dürfen**; Da der Quelltext vorliegt, kann jeder interessiert Entwickler das Programm beliebig erweitern, verbessern und den individuellen Bedürfnissen anpassen. Fehler und Sicherheitslücken können durch die Mitarbeit von Programmieren in aller Welt schnell aufgespürt und behoben werden. Kein kommerziell orientiertes Unternehmen könnte eine vergleichbare große Zahl von Entwicklern bezahlen und so schnell reagieren.
- **Das Recht, die Open-Source-Software sowie alle Änderungen und Verbesserungen am Quellcode weiterzugeben**. Jeder Anwender kann Änderungen am Quellcode vornehmen und diese weitergeben. Dadurch wird die Qualität der Software ständig verbessert.
- **keine Lizenz-Gebühren**; Für Unternehmen und Behörden mit ihren häufig knappen Budgets für EDV-Ausrüstung beinhaltet Open-Source-Softeware den Vorteil, dass weder für das zugrundeliegende Betriebssystem noch für die Verbesserungen oder Änderungen an der Software Lizenzgebühren erhoben werden dürfen. (Exkurs: Dadurch hätten einzelne Unternehmen bei der Behebung von Jahr-2000-Problemen Kosten in Millionenhöhe einsparen können.)
- **Lösung von Software-Problemen**; Open-Source-Software ist natürlich kein Allheilmittel, kann aber alleine oder in Kombination mit kommerziellen Programmen zahlreiche Software-Probleme lösen. So läuft Apache, der Web-Server mit dem offenen Quellcode, auch auf Windows NT. Mit dem von dem Australier Andrew Trigell entwickelte Open-Source-Produkt Samba kann ein Linux-System mit einem Windows-Rechner verbunden und so beispielweise als Datei- oder Druckerserver für Windows 9x/NT Arbeitsplätze verwendet werden.
- **große Entwickergemeinde**; prominentes Beispiel ist Samba: Die im Februar 1999 erschienene Version 2.0.1 enthielt einen gravierenden Fehler. Innerhalb weniger Stunden war dieser Bug behoben und die Version 2.0.2 von Samba stand im Internet zur Verfügung.

- die Software kann von firmenexternen Programmieren und Anwendern gezielt überprüft, abgeändert und weiterentwickelt werden
- durch offengelegten Quellcode läuft die Software in der Regel verlässlich, stabil, kostengünstig und sicher
- **Keine Exklusivrechte an der Software**. Open-Source-Software steht allen offen. Dadurch kann weder ein einzelner Programmierer, noch ein Unternehmen die Richtung der Entwicklung vorgeben. Auch die Probleme, die bei Anbietern kommerzieller Software entstehen, wenn diese ihre Geschäftstätigkeit aufgeben oder von einer anderen Firma übernommen werden, gibt es die Open-Source-Software nicht, weil ihre Entwicklung und ihr Fortbestehen nicht einzelnen Firmen abhängt. Stellt eine Entwicklungsgruppe ihre Arbeit ein, kann diese von anderen aufgenommen werden.
- **Große Gebietsabdeckung**; Kaum ein Hersteller ist in der Lage, Software anzubieten, die so ein weites Gebiet abdeckt, wie jene im Bereich der offenen Quellcodes. Dazu sind die Open-Source-Programme auch preiswerter als kommerzielle Produkte. So stehen etwa für GNU/Linux rund 1.000 Applikationen von der Datensicherung über Serverfunktionen bis hin zu allen Internetdiensten kostenlos zum Download zur Verfügung. Eine uneingeschränkte Nutzerzahl kann – gleichgültig, ob für den privaten oder kommerziellen Einsatz – auf sämtliche Anwendungen zugreifen.
- **Unabhängig von Hardwareplattform**; Bei der Auswahl der Hardware bleiben die Nutzer weitgehend ungebunden. Die Open-Source-Software läuft auf vielen Systemen, das sie oft in weit höherem Maße als die meiste proprietäre (Proprietät = Eingentums(recht); [lat.,frz.]) Software auf die Hardware angepasst werden kann. Die Verwendung offener Standards schafft die Voraussetzung für Kompatibilität und ermöglicht die Portierung auf andere Hard- oder Software-Plattformen.
- üblicherweise wird Open Source Software erst dann eine Stable Release (stabile Version) herausgeben, wenn das Programm ausgiebig getestet wurde und von den Entwicklern als stabil betrachtet wird

vorerst wird eine Experimental Release herausgeben, bei dem jeder Nutzer weiß, das es sich um eine Software handelt, deren Entwicklung noch nicht abgeschlossen und somit auch weniger verlässlich ist (weniger gewinnorientiert)

Schwächen und Nachteile von Open-Source-Software

- **Anwendungssoftware auf Arbeitsplatzrechnern**; Für manches Einsatzgebiet, wie beispielsweise Buchhaltung und Rechnungswesen, Projektmanagement, Workgroup-Management und verschiedene Bereiche der Branchen-Software gibt es derzeit noch keine ausgereiften Open-Source-Produkte.
- **Im- und Export den Dokumenten**; Die Open-Source-Programme verfügen zwar über einen Filter, mit denen Dokumente, die beispielsweise mit MS-Office-Paketen erstellt wurden, eingelesen und anschließend wieder im MS-Datei-Formaten abgespeichert werden können. Allerdings funktioniert der Im- und Export von Textdokumenten mitunter nicht zuverlässig oder nur eingeschränkt. Schwieriger wird der Dokumentenaustausch bei komplexen Tabellenkalkulationen. Diese enthalten häufig Makros – das sind Sammlungen von Befehlen, die einer Zelle in einer Tabellenkalkulation zugeordnet sind. Oft scheitern die Filter an den Makros.
- **Hardware-Unterstützung**; In manchen Fällen weist die Unterstützung, z.B. bei hardware-beschleunigten Grafikkarten oder Mulitmedia-Equipment, wie Scanner, Mängel auf. Ebenso ist mit Schwierigkeiten zu rechnen, wenn ein nicht postscript-fähiger Drucker mit Open-Source-Software betrieben werden soll.
- **hohe Anforderungen an den Nutzer**; Der Umgang mit Open-Source-Betriebssystemen und Anwendungsprogrammen stellt im allgemeinen höhere Anforderungen an die Kenntnisse des

Nutzers über die Funktionsweise und den Aufbau des Systems als etwa im Microsoft-Umfeld üblich.

- **Beschaffung von Informationen**; Für Neueinsteiger im Open-Source-Bereich kann auch die Beschaffung von Informationen, welche Software für einen bestimmten Anwendungszweck in Frage kommen würde, zum Problem werden. Bis vor einigen Jahren waren die meisten Open-Source-Produkte nur Insidern bekannt. Trotz des zunehmenden Bekanntheitsgrades in der Öffentlichkeit ist es mitunter noch mühsam herauszufinden, ob für bestimmte Anwendungsbereiche bereits Open-Source-Lösungen angeboten werden. Das Projekt BerliOS, gegründet vom Forschungsinstitut für offenen Kommunikationssysteme (FOKUS) des GMD-Forschungszentrums für Informationstechnik, will für Abhilfe sorgen.
- zu umständliche Konfiguration
- zu wenig Benutzerfreundlichkeit der grafischen Oberfläche

(- durch den Einsatz freier grafischer Benutzeroberflächen wurde dieses Problem schnell behoben)

Unterschiede zu Free Software, Public Domain, Freeware, Shareware

	Kurzbeschreibung	Quellcode ist modifizierbar	Modifikationen müssen immer unter denselben Bedingungen veröffentlicht werden	Produkt ist lizenzgebührenfrei	Nutzung ist uneigeschränkt	Weiterverteilung ist erlaubt
Open-Source-Software	Quelloffene Software soll Unternehmen und Wirtschaft näher gebracht werden. Die kommerzielle Nutzung soll einfacher sein (im Vergleich zu Free Software)	x		x	x	x
Free Software	Der Quellcode ist offen, und seine Modifikationen müssen auch offen bleiben.	x	x	x	x	x

	Kurzbeschreibung	Quellcode ist modifizierbar	Modifikationen müssen immer unter denselben Bedingungen veröffentlicht werden	Produkt ist lizenzgebührenfrei	Nutzung ist uneigeschränkt	Weiterverteilung ist erlaubt
Open-Source-Software	Quelloffene Software soll Unternehmen und Wirtschaft näher gebracht werden. Die kommerzielle Nutzung soll einfacher sein (im Vergleich zu Free Software)	x		x	x	x
Public Domain	Diese Software ist als ein Sonderfall zu betrachten: Der Urheber verzichtet komplett auf das *copyright*. Somit wird diese Software zum Gemeingut und kann uneigeschränkt genutzt werden. Sollte der Quellcode zur Verfügung stehen, liegt Open-Source-Software vor.	x		x	x	x

	Kurzbeschreibung	Quellcode ist modifizierbar	Modifikationen müssen immer unter denselben Bedingungen veröffentlicht werden	Produkt ist lizenzgebührenfrei	Nutzung ist uneigeschränkt	Weiterverteilung ist erlaubt
Open-Source-Software	Quelloffene Software soll Unternehmen und Wirtschaft näher gebracht werden. Die kommerzielle Nutzung soll einfacher sein (im Vergleich zu Free Software)	x		x	x	x
Freeware	Diese Art der Software ist keine Free Software. Es werden zwar keine Lizenzgebühren erhoben, aber der Quellcode steht nicht zur Verfügungen.			x	x	x

	Kurzbeschreibung	Quellcode ist modifizierbar	Modifikationen müssen immer unter denselben Bedingungen veröffentlicht werden	Produkt ist lizenzgebührenfrei	Nutzung ist uneigeschränkt	Weiterverteilung ist erlaubt
Open-Source-Software	Quelloffene Software soll Unternehmen und Wirtschaft näher gebracht werden. Die kommerzielle Nutzung soll einfacher sein (im Vergleich zu Free Software)	x		x	x	x
Shareware	Hierunter wird Software verstanden, die für eine vom Autor festgelegte Testphase genutzt werden darf. Ist die Testphase abgelaufen, so sind Lizenzgebühren zu bezahlen.					x

Motivation

Hier sind nicht die Softwarefirmen, sondern die Softwareentwickler die Aktoren, die die Existenz der Open-Source-Software möglich machen.
Empirische Untersuchungen zu OSS bestätigen die Existenz der Motivationen, aber es fehlt die Abschätzungen über die Relevanz dieser verschiedenen Beweggründe.
Um die Begriff „Motivation“ besser zu verstehen, ist eine Änderung des Blickwinkels erforderlich. OSS ist nicht aus der wirtschaftlichen Sichtweise der Softwarefirmen zu betrachten. Im Zentrum stehen die Software-Benutzer bzw. –Programmierer oder ’Prosumer’, wie sie A. Toffler in seinem Buch „The Third Wave“ (1981) bezeichnet. Prosumer sind Benutzer, welche die Software an ihre Bedürfnisse anpassen und weiterentwickeln.
Das Handeln der Prosumer ist aber nicht weniger rational als das der Softwarefirmen. Quellcode wird nur offen gelegt, wenn der Nutzen der Offenlegung größer ist als die Kosten einer solchen Handlung

Gebrauch

Der wahrscheinlich einfachste Grund, warum sich der Softwareentwickler für ein OS-Projekt engagiert, ist der, dass er die von diesem Projekt erzeugte Software selber gebrauchen kann. Entweder wird ein Problem mit geeigneter Software gelöst oder eine bestehende Software wird den Bedürfnissen angepasst.

Reputation

Eric S. Raymond beschrieb in seinem Essay Homesteading the Noosphere (2000) die Normen und Tabus der OS-Bewegung in Bezug auf den Erwerb von Reputation. Raymond zeigt auf, dass „code forking“ (aufsplitten der Code-Basis in inkompatible Versionen) und besonders das Löschen der Namen der Beitragsleister aus den „credit files“ der Applikationen streng verpönt ist.
Ohne diese Normen wäre es schwierig nachzuvollziehen von wem welcher Beitrag kommt und der Aufbau von Reputation wäre stark behindert.

Signalproduktion

J. Lerner und J. Tirole „The Simple Economics of Open Source“ (2001) analysierten im Sinn der Reputation unter dem Aspekt der Signalproduktion. Durch die geschaffene Norme ist es möglich die Beiträge des einzelnen Entwicklers genau verfolgen zu können. Der Status eines Entwicklers ist ein genaues Abbild seiner Reputation, das sehr transparent die Qualität und Quantität der Beiträge eines Entwicklers reflektiert.
Wenn die Reputation als gültiger Indikator für das Talent eines Entwicklers ist, kann die Reputation als Signal für einen potentiellen Arbeitgeber wirken, der somit gültiger Rückschlüsse auf diese Talent schließen kann. Diese Signalproduktion wirkt am stärksten, wenn die technische Herausforderung groß, wenn die relevante Öffentlichkeit (d.h. die Peergruppe) technisch erfahren ist, zwischen guten und herausragenden Leistungen unterscheiden sowie Leistung und Können wertschätzen kann (Steven Weber „The Political Economy of Open Source Software“ (2000) und E. Franck / C. Jungwirth „Die Governance von Open-Source-Projekten“ (2002b)).

Identifikation

Ist eine Person gut eine Gruppe involviert und kann sich mit deren Zielen identifizieren, so ist häufig ein großes Engagement dieser Person für diese Ziele zu beobachten. G. Hertel / S. Niedner / S. Herrmann „Motivation of Software Developers in Open-Source-Projects“ (2003) haben nachweisen können, dass sich durch die Identifikation mit der Entwicklergemeinde ein signifikanter Anteil des Engagements der Entwickler erklärt werden kann.

Lernen

Natürlich haben die Softwareentwickler auch den Wunsch ihre Fähigkeiten zu verbessern (dies wies R.A. Ghosh in seiner FOSS-Studie (Free and Open Source Software) nach). Aber auch der Lernaspekt scheint keine unwesentliche Rolle zu spielen, denn der Einsatz von

neuster Technologie zur Bewältigung eines komplexen Problems ist für alle Beteiligten mit einem Gewinn an Erfahrung gebunden.

Altruismus

Das Engagement in Open Source kann auch mit dem Gefühl für das Wahre und Richtige begründet werden. Die Freiheit der Menschen hängt mit quelloffener Software zusammen, aber diese Gut ist durch die kommerzielle und proprietäre Softwareanbieter gefährdet. Diesen Standpunkt vertritt auch Stallman und seine FSF.

Nach K. Lakhani / R. Wolf „Why Hackers Do What They Do: Understanding Motivation Effort in Free / Open Source Software Projects“ (2003) ist dieser Grund für ein Drittel der Entwickler relevant.

Spaß

Weiter empfinden Programmierer Spaß an ihrer Tätigkeit. Linus Torvalds betitelt seinen Rückblick auf die Entstehung von Linux als „Just for fun“, wobei die Freizeit ein nicht unbedeutender Faktor ist.

In der FASD-Studie (Fun and Software Development) wurde überprüft:

Je mehr Spaß das Programmieren macht und je mehr Freizeit der Softwareentwickler hat, desto größer ist sein Einsatz beim Entwickeln von OSS.

Aber auch Spaßsucher sind rational denkende und handelnde Personen. Auch wenn das Programmieren Spaß macht, ist es noch besser mit dieser Tätigkeit Geld zu verdienen, als das Resultat zu verschenken.

Um die Bedeutung von „Spaß“ besser zu verstehen, ist es notwendig die Unterschiede zum Programmieren in einem OS-Projekt und der Tätigkeit in einem kommerziellen Unternehmen aufzuzeigen:

- 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 OS-Projekt hat keine formale Autorität.
- Die Programmierer in einem OS-Projekt können üblicherweise nicht über direkte monetäre Anreize zu einem Engagement oder zur Steigerung ihres Engagements bewegt werden.
- Ein OS-Projekt hat üblicherweise keine Abgabetermine.

Diese Differenzierung macht klar und nachvollziehbar, dass ein OS-Entwicklungsmodell mehr Spaß zulässt.

In einer Studie von Lakhani und Wolf (2003) wurde Spaß als Motivation weitgehend bestätigt.

Empirische Studien zur Motivation

Studie und Thema	Methode	Sample-Größe
Hars und Ou (2001) Motivation von Open-Source-Entwicklern	Online-Fragebogen	81

Ergebnisse:

80% der Befragten bewerteten **Selbstbestimmung** als hoch bis sehr hoch
88% war die **Bildung von Humankapital** wichtig

93% ist **Selbstbestimmung** für die **OS-Entwickler aus dem IT-Bereich**, die in ihrer Freizeit OS entwickeln **überdurchschnittlich** motivierend
61,5% ist **Selbstbestimmung** für **bezahlte OS-Programmierer** nur **unterdurchschnittlic**h motivierend

97% ist für **Studierende und Hobbyprogrammierer der Lerneffekt überdurchschnittlich** wichtig

A. Hars / S. Ou „Working for Free? – Motivation of Participating in Open-Source-Projects" (2001)

Studie und Thema	Methode	Sample-Größe
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 bewerte.
In der Varianzanalyse erhielt das Motiv „Identifikation mit der Projektgruppe" mit 22,9 % den höchsten Erklärungsgehalt.
Das Ergebnis ist statistisch signigikant.

G. Hertel / S. Niedner / S. Herrmann „Motivation of Software Developers in Open-Source-Projects" (2003)

Studie und Thema	Methode	Sample-Größe
Lakhani und Wolf (2003) Motivation von Open-Source-Entwicklern	Online-Fragebogen	648

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:

59% Gebrauch, 45% Spaß, 41% Lernen, 33% Altruismus

K. Lakhani / R. Wolf „Why Hackers Do What They Do: Understanding Motivation Effort in Free / Open Source Software Projects“ (2003)

(Flow -> der Handlungsablauf wird als glatt erlebt. Ein Schritt geht flüssig in den nächsten über, als liefe das Geschehen gleitend wie aus einer inneren Logik)

Quellen

- Bernd Lutterbeck, Robert A. Gehring, (Hrsg.) Open Source Jahrbuch 2004: Zwischen Softwareentwicklung und Gesellschaftsmodell, Berlin (Lehmanns Media) 2004, ISBN: 3-936427-78-X,
  LINK: http://www.think-ahead.org/
- Open Source: Leitfaden für kleine und mittlere Unternehmen:
  LINK: http://oss-broschuere.berlios.de/broschuere/broschuere-de.html
- Open-Source-Definition:
  LINK: http://de.wikipedia.org/wiki/Open-Source-Lizenz