

Open Source – Ansätze für ein neues, globales Innovationsmodell?

Geschichte:

Bis Mitte der sechziger Jahre war es gängige Praxis, die Quellcodes seiner entwickelten Software für Jedermann zugänglich zu machen. Ob Hobby- oder Profi-Programmierer, jeder, der Interesse hatte, konnte die Software beliebig nutzen und verändern. Computer-Hersteller verdienten damals ihr Geld mit dem Verkauf von Hardware, ihre Software wurde als Zubehör mitgeliefert.

1965 jedoch bot IBM erstmals seine Software ohne den dazugehörigen Quellcode an. Das am Anfang noch kleine Unternehmen hatte sich zu einer großen Firma entwickelt, die Mithilfe unternehmensfremder Programmier bei der Weiterentwicklung der eigenen Software war nicht mehr notwendig. IBM erkannte schnell, dass Software eine riesige Einnahmequelle werden würde, die geschützt werden muss. Es wurden Lizenzverträge entworfen, die eine Weitergabe von Software einschränkten oder ganz verboten. Somit wurden Programm-Quellcodes von 1965 bis 1980 zu den meist geschützten Geheimnissen der Software-Entwickler. User dieser Software waren von diesem Zeitpunkt an auf die Unternehmen angewiesen, wenn Fehler in Programmen auftraten oder andere Komponenten erweitert oder verändert werden mussten.

1969 entstand die erste Version von Unix. Hier sahen die Entwickler eine besonders große Einnahmequelle, da es sich um ein Betriebssystem handelte. Deswegen wurde auch hier der Quellcode nicht veröffentlicht.

Richard Stallman, ein Student vom *Massachusetts Institute of Technology*, war über diese Entwicklung der Software-Gemeinde sehr unglücklich. Deshalb gründete er 1984 das „GNU“-Projekt, was sich zum Ziel setzte, ein Unix ähnliches Betriebssystem zu entwickeln, dessen Quellcode jedoch frei zugänglich war. Stallman wollte damit die offene Zusammenarbeit bei der Softwareentwicklung, wie sie Anfang der siebziger Jahre noch vorhanden war, erneut herstellen. Nutzen daraus sollten alle Computer-Anwender ziehen. Sämtliche Programme sollten ohne Einschränkungen zur freien Verfügung stehen, weitergegeben und weiterentwickelt werden dürfen. Der Erfolg war beachtlich: Immer mehr Programmierer schlossen sich dem GNU-Projekt an und entwickelten ein komplettes Betriebssystem, inkl. einiger Tools und Anwendungssoftware. Das einzige was fehlte, war ein Kernel. Diesen stellte 1991 der Finne Linus Torvalds zur Verfügung. So entstand das erste, komplett quell offene Betriebssystem GNU/Linux.

Richard Stallman, manchmal als „Vater von GNU“ bezeichnet, vertritt die Meinung, dass „frei“ für eine Software heißt, dass sie uneingeschränkt nutzbar sei. Diese Meinung vertritt nicht nur Stallman, sondern auch seine Mitstreiter am *Massachusetts Institute of Technology*. Gemeinsam gründeten sie die „Free Software Foundation“ (FSF), die bis heute vehement diese Meinung vertritt und GNU-Projekte unterstützt. Hierzu entwickelte Richard Stallman die *GNU General Public License*, welche die Freiheit von Software schützt. Derzeit sind knapp 3000 GNU-Pakete im „Free Software Directory“ eingetragen.

Eric S. Raymond, Bruce Perens und Tim O'Reilly gründeten 1998 die „Open Source Initiative“ (OSI). Diese schlug vor, Software mit offenem Quellcode ab diesem Zeitpunkt als „Open Source-Software“ zu bezeichnen. Um der Wirtschaft ihre Skepsis zu nehmen, erstellte die OSI an die Wirtschaft angepasste Lizenzen.

Definition:

Damit eine Software sich „Open Source“ nennen darf, muss sie einige Kriterien erfüllen:

- Der Quelltext muss frei verfügbar sein und in einer für den Menschen lesbaren Form vorliegen (z.B. als Quelltext eines C++ oder Java-Programms).
- Die Software darf beliebig genutzt und kopiert werden, es gibt keine Nutzungsbeschränkungen.

Die Software (auch Teile davon) darf verändert und in der veränderten Form weitergegeben werden.

„Open Source“ steht also für Software, die frei genutzt werden darf. Das „frei“ darf man hier nicht falsch verstehen. Es steht im Sinne von „Freiheit“, nicht „kostenlos“ („Free as free speech, not as free beer.“). Denn dass eine Software „open source“ ist, impliziert nicht, dass sie auch kostenlos ist. Besonders im Englischen, wo die Bezeichnung „free software“ lautet, gibt es deshalb oft Missverständnisse. Verstärkt wird dies durch die Tatsache, dass die meisten Open-Source-Programme tatsächlich kostenlos verfügbar sind. Eine Weitergabe von (veränderter) Open-Source-Software kann aber auch gegen eine Gebühr erfolgen. Als Beispiel sei hier die Firma Red Hat zu nennen, die zu dem Open Source-Betriebssystem „Linux“. Aber auch wenn man Open-Source-Software käuflich erwirbt, gelten die oben genannten Rechte. Man darf auch dann die Software verändern und kostenlos weitergeben. Meistens erwirbt man auch nicht die Software an sich, sondern Supportleistungen und/oder Dokumentationen rund um die Software.

Lizenzen:

Es gibt sehr viele Open Source-Lizenzen. Wir gehen hier nur auf die vier verbreitesten ein.

- GNU General Public Licence (GNU GPL):
Dies ist die am weitesten verbreitete Open Source-Liezenz. Sie besagt, dass die unter ihr veröffentlichte Software den oben genannten Open Source-Kriterien entspricht und außerdem, dass sämtliche veränderte Software wieder unter der GPL veröffentlicht werden muss. So soll verhindert werden, dass Open Source-Software zur Basis für kommerzielle Software wird („Copyleft“-Verfahren). Diese Eigenschaft der GPL steht stark in der Kritik, da sie die freie Wahl der Lizenz für ein Programm verhindert.
- GNU Lesser General Public Licence (GNU LGPL):
Die LGPL besagt im Prinzip dasselbe wie die GPL, mit der Ausnahme, dass unter der LGPL veröffentlichte Software nicht unbedingt wieder unter der LGPL erscheinen muss, wenn sie verändert wird. Ursprünglich hieß diese Lizenz auch „Library GPL“ und wurde entworfen, um es großen Firmen zu ermöglichen, Programmbibliotheken, die unter der GPL veröffentlicht waren, in ihre Programme einzubinden (ansonsten hätten sie diese Programme ja auch unter der GPL veröffentlichen müssen).

- GNU Free Document Licence (GNU FDL):
Auch diese Lizenz ähnelt der GPL, mit dem Unterschied, dass sie nicht für Software, sondern für Dokumente gilt. Entworfen wurde die FDL, um die Dokumentationen zum GNU-Projekt eine ähnliche Freiheit wie den Programmen zu bieten. Auch die freie Wissensdatenbank *Wikipedia* nutzt diese Lizenz.
- Berkely Software Distribution-Lizenz (BSD-Lizenz):
Die Regeln der BSD-Lizenz sind nicht ganz so streng wie die der GPL. Die Programme gelten zwar als Open Source und erfüllen auch alle Kriterien, jedoch muss der Quelltext eines modifizierten Programms nicht veröffentlicht werden. Nur ein Copyright-Vermerk auf den ursprünglichen Autor muss beibehalten werden. Diese Lizenz eignet sich daher gut für kommerzielle Produkte.

Lizenzen: Open Source-Defintion

Die Open Source-Definition (OSD) ist eine Richtlinie zur Bewertung von Open Source-Lizenzen. Sie wurde 1997 von Bruce Parens entwickelt um genauer zu definieren, was „Freiheit“ ist. Zunächst wurde diese Definition nur für das Debian-Projekt genutzt. Eric S. Raymond stellte aber schnell fest, dass dieser „Debian Free Software Guideline“, wie es damals noch hieß, das richtige Dokument ist, um Open Source-Software zu definieren. Er beauftragte Parens, die Debian-spezifischen Absätze zu entfernen, „free software“ gegen „open source“ auszutauschen und den Namen zu ändern. So entstand die Open Source-Definition.

Lizenzen: Copyleft

Copyleft ist ein Schutzverfahren in den Lizenzen der freien Software, das einen bestimmten Aspekt des Copyrights in sein Gegenteil zu verkehren versucht (daher auch der Name). Laut dem Urheberrecht hat der Autor eines Werkes ein Mitspracherecht, was mit diesem Werk geschieht. War ein Werk aber vorher frei kopierbar bzw. veränderbar, so überträgt sich dieses Recht nicht automatisch auf die Bearbeitung. Copyleft versucht genau diesen Aspekt umzukehren, denn unter einer Copyleft-Lizenz veröffentlichte Werke bleiben immer frei kopierbar bzw. änderbar. Auf diese Art und Weise wird verhindert, dass Open Source-Software zur Basis von kommerzieller Software wird.

Beispielsoftware für Open Source:

Open Source-Software gibt es in allen erdenklichen Softwarebereichen.

Betriebssysteme:	Debian GNU/Linux
Anwendungssoftware:	Open Office, Emacs, Mozilla Firefox
Entwicklungssoftware:	Perl, PHP
Serverdienste:	Apache, Samba
Datenbanken:	MySQL
Desktop/X-Window-Systeme:	KDE, Gnome

Innovationsmodell Open Source-Software:

Betrachtet man die Open Source-Entwicklung als ein neues Innovationsmodell gegen die herkömmliche Softwareentwicklung, welche die Software durch private Eigentumsrechte schützt, muss man sich drei zentrale Fragen stellen.

1. Warum sind private Eigentumsrechte an Innovationen nicht immer effizient?
2. Warum tragen Menschen freiwillig zu einem öffentlichen Gut bei?
3. Wie muss die Kooperation zwischen kommerziellen Firmen und nicht kommerziellen Gemeinschaften ausgestaltet sein?

Zu 1.: In der Regel werden Innovationen vorangetrieben um damit Geld zu verdienen. Aus diesem Grund werden Patente auf neue Entwicklungen angemeldet, damit kein zweiter oder sogar dritter die Möglichkeit hat, Profit aus dieser Innovation zu schlagen. Durch diese Eigentumsrechte entsteht eine eingeschränkte Möglichkeit die Entwicklung voranzutreiben, und es kommt in der Praxis häufig zu einer Unternutzung der Innovation (Blockierung). Eine solche Blockierung tritt zum Beispiel dann auf, wenn viele einzelne Parteien Rechte an der Entwicklung (Innovation) besitzen. Jede neue Erweiterung müsste mit allen Parteien, die Rechte haben, abgesprochen werden, und entsprechende Verträge geschlossen. Dieser Umstand ist sehr teuer und macht das Vorantreiben der Innovation nicht mehr vorteilhaft.

Zu 2.: Bei der Motivation, an der Entwicklung von Open Source-Software teilzunehmen, unterscheidet man zwei unterschiedliche Bedürfnisbefriedigungen.

1. direkte Bedürfnisbefriedigung (intrinsisch): Das bedeutet, das die Programmierer aus Spaß am Programmieren die Entwicklung der Open Source-Software durchführen. Die Aktivität wird ihrer selbst willen geschätzt und wird auch ohne Belohnung ausgeführt. Die entstandenen Programme werden meist zum Eigenbedarf erstellt, oder es werden an vorhandenen Produkten Weiterentwicklungen implementiert.
2. indirekte Bedürfnisbefriedigung (extrinsisch): Dies bedeutet, das Bedürfnisbefriedigung nicht durch das Programmieren erlangt wird, sondern zum Beispiel aus dem Verkauf des fertigen Produktes. In diesem Fall würde die Bedürfnisbefriedigung das Geld sein welches aus dem Verkauf erzielt wird. Eine andere indirekte Motivation wäre die Verbesserung der eigenen Reputation. Durch das entwickeln der Open Source Software erhofft man sich eventuell einen besseren Job oder höheres Gehalt, welches dann die Befriedigung darstellen würde.

Zu 3.: Es muss eine Ausgewogenheit zwischen intrinsischer und extrinsischer Motivation bestehen. Voraussetzung ist aber, dass beide Motivationen vorhanden sind. In der Anfangszeit von Open Source-Software gab es ausschließlich Programmierer mit einer direkten Bedürfnisbefriedigung. Daraus resultierte das die Software die entstanden ist, schwierig zu bedienen war. Sie wurde ausschließlich von Experten für Experten gemacht. Erst seitdem kommerzielle Firmen sich mit Open Source-Software befassen, und hierdurch die indirekte Bedürfnisbefriedigung hinzugekommen ist, sind die Programme benutzerfreundlicher geworden und für jedermann zu benutzen.

Heutzutage gibt es eine Ausgewogenheit zwischen intrinsischer und extrinsischer Motivation, wobei kommerzielle Unternehmen stark darauf bedacht sind, die Normen und Werte der Programmierer mit intrinsischer Motivation zu wahren und einzuhalten! Regelverstöße sind oft schwer nachzuweisen, da die „Täter“ schwer zu identifizieren sind. Auf der anderen Seite sind auch Programmierer mit direkter Bedürfnisbefriedigung auf die kommerziellen Firmen angewiesen, da erst hierdurch die weite Verbreitung der Software an normale Konsumenten sichergestellt wird.

Vorteile von Open Source-Software:

- stabil laufende Software
- Benutzerfreundlichkeit wird zunehmend gesteigert
- Verfügbarkeit des Quellcodes und das Recht, ihn ändern zu dürfen
- Das Recht, die Open Source-Software sowie alle Änderungen und Verbesserungen am Quellcode weiterzugeben
- Keiner hat Exklusivrechte an der Software
- Läuft auf vielen Systemen, da sie schneller und besser auf Hardware angepasst werden kann
- Wartungsfreundliche Software, d.h. Personal wird in Unternehmen gespart
- bei großen Virus-Attacken, wie sie in letzter Zeit vermehrt auftreten, ist Open Source-Software eher wenig bis gar nicht betroffen, Open Source-Programme sind zudem besser vor trojanischen Pferden o.ä. geschützt als Software, deren Quellcode das Geheimnis des Herstellers bleibt. Durch Offenlegung des Quellcodes lässt sich überprüfen, ob das Programm Hintertüren enthält, die das Ausspionieren des Systems ermöglichen. Diese können dann auch sofort geschlossen werden.

Nachteile von Open Source-Software:

- Hauptanteil der Probleme mit Open Source-Software betrifft die Anwendungssoftware an Arbeitsplätzen
- Es existieren zwar Converter, mit denen Dokumente aus z.B. MS-Office geöffnet und wieder abgespeichert werden können, allerdings funktioniert der Im- und Export von Textdokumenten mitunter nicht zuverlässig oder nur eingeschränkt
- Die Hardware-Unterstützung weist in manchen Fällen, z.B. bei Hardwarebeschleunigten Grafikkarten oder bei Multimedia-Equipment, wie z.B. Scannern, Mängel auf. Ebenso ist mit Schwierigkeiten zu rechnen, wenn ein nicht PostScript fähiger Drucker mit Open Source-Software betrieben werden soll.
- Der Umgang mit Open Source-Betriebssystemen und Anwendungen stellt im Allgemeinen höhere Anforderungen an die Kenntnisse des Nutzers über die Funktionsweise und den Aufbau des Systems als es etwa im Microsoft-Umfeld üblich ist. 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. Allerdings hat sich in diesem Bereich in letzter Zeit viel getan.