mehr zum thema:
www.sap.com/company/research
www.riehle.org

GELD VERDIENEN MIT OPEN-SOURCE

Dieser Artikel analysiert die ökonomischen Positionen von Systemintegratoren, Open-Source-Service-Firmen und Softwareentwicklern. Besonderes Gewicht wird dabei auf die Interaktion dieser ökonomischen Perspektiven gelegt und darauf, was sie für die Karriere von Softwareentwicklern in der neuen Open-Source-Welt bedeutet.

der autor

Dr. Dirk Riehle (E-Mail: dirk@riehle.org) leitet die Open-Source-Forschungsgruppe von SAP Labs, LLC, in Palo Alto, Kalifornien. Ihn interessieren die technischen, ökonomischen, und strategischen Aspekte von Open-Source im Unternehmenseinsatz.

Anwender lieben Open-Source-Software, schließlich hilft sie zumeist, Geld zu sparen. Was aber ist mit Softwareentwicklern, die bisher mit proprietärer Software ihren Lebensunterhalt verdient haben? Was bleibt, wenn scheinbar alle relevante Software umsonst zu haben ist? Dieser Artikel betrachtet die Veränderungen, die durch Open-Source im Softwaregeschäft entstanden sind, aus drei Perspektiven: aus der der großen Systemintegratoren, aus der der meist mittelständischen Softwarefirmen sowie aus der von Angestellten, den individuellen Softwareentwicklern. Er diskutiert, wie sich neue Geschäftsmodelle herausgebildet haben, die es diesen Marktteilnehmern ermöglichen, trotz und gerade mit Open-Source Geld zu verdienen. Alle drei Perspektiven sind wichtig, weil sie ineinander greifen und voneinander abhängig sind. Besonderes Gewicht wird auf die Perspektive des *Open-Source-Committers* gelegt und darauf, welche Bedeutung Open-Source für Entwicklerkarrieren haben kann.

Open-Source ist die schöne neue Welt der Softwareentwicklung. In nur 15 Jahren hat es Open-Source-Software geschafft, zu einem zentralen Bestandteil der Informationstechnologie-Welt zu werden. Open-Source-Software ist Software, die zumeist kostenfrei im Internet bereitgestellt wird und die häufig von so hoher Qualität ist, dass sie mit kommerziellen Softwarepaketen konkurrieren kann.

Open-Source-Software wird in weiten Teilen den Arbeiten von Richard Stallman („freie Software", GNU-Lizenzen, vgl. [FSF]) und Linus Torvalds, Schöpfer des Unix-Betriebssystems Linux (vgl. [OSDL]), zugeschrieben. Heutzutage handelt es sich aber um ein florierendes „Ökosystem", bestehend aus unterschiedlichsten Teilnehmern: kommerziellen Firmen, eingetragenen Vereinen („non-profits"), freien Softwareentwicklern, Anwendern und anderen. Wichtig bei Open-Source ist, dass die Software unter einer bestimmten Lizenz zur Verfügung gestellt wird, die ihre Benutzungsmöglichkeiten festlegt. Die Lizenz bestimmt weiterhin, was mit Veränderungen zu geschehen hat, die ein Anwender an der Software vornehmen mag. Die *GNU Public License* zum Beispiel erzwingt im Prinzip, dass Veränderungen und Erweiterungen am Quelltext allen daran Interessierten bereitgestellt werden müssen. Im Vergleich dazu erzwingt die *Apache-Lizenz* dies nicht, sondern überlässt es dem Anwender, ob er seinen Quelltext beitragen möchte oder nicht.

Betrachtet man den Markt, so stellt man mit Erstaunen fest, dass viele große IT-Firmen, allen voran IBM, aktiv an Open-Source-Softwareprojekten mitarbeiten. Dies sollte auf den ersten Blick verwunderlich sein, da IBM als großer Softwarehersteller sich doch ins eigene Fleisch zu schneiden scheint, wenn es Open-Source-Software als Alternative zu den eigenen Produkten unterstützt. Ein Beispiel ist die Unterstützung von Geronimo, einem J2EE Application Server, der als Konkurrenzprodukt zu IBMs eigenem „Websphere" gelten kann. Schaut man genauer hin, erkennt man aber verschiedene Mechanismen, warum IBM schlussendlich doch von Open-Source-Software profitiert: Zum einem geschieht dies durch komplementäre Produkte und zum anderen durch die Anbindung von Kunden über Open-Source an die eigene Plattform und den späteren „Upsell", also den Verkauf proprietärer Produkte. Der folgende Abschnitt diskutiert die Perspektive von IBM und anderen Systemintegratoren.

Open-Source-Softwareprojekte werden aber nicht nur von Systemintegratoren betrieben, sondern auch von kleinen Softwarefirmen, die sich zum Teil ausschließlich auf dieses eine Projekt spezialisieren und es als Produkt vertreiben. Zumeist bieten diese Firmen Support und Implementierungshilfen. Auch hier stellt sich die Frage, warum diese Firmen nicht lieber eine proprietäre Software entwickeln und vertreiben. Ein einfache Antwort ist, dass Kunden lieber eine kostenfreie Open-Source-Software verwenden als eine proprietäre Lösung. Und darüber ergibt sich ein neuer Markt für Firmen, der sie dazu bewegen kann, zu einem Open-Source-Softwareprojekt beizutragen und darauf basierend kommerzielle Dienste anzubieten. Der Abschnitt „Die Sicht der Softwarefirmen" diskutiert die Sicht der zumeist kleinen Softwarefirmen, die sich auf ein Produkt spezialisieren.

Am Anfang der Lieferkette aber stehen die Softwareentwickler. Von ihnen ging Open-Source-Software aus und sie sind weiter dabei, Open-Source-Software zu entwickeln. Vielfach geschieht dies aus Begeisterung darüber, zu einer Gemeinschaftsarbeit beizutragen, die anderen einen signifikanten Nutzen bringt. Aus ökonomischer Sicht ignorieren wir in diesem Artikel aber die altruistische Motivation und konzentrieren uns auf die ökonomische: Laut einer Untersuchung verdienen nämlich die *Committer* an Open-Source-Projekten mehr als ihre Nicht-*Committer*-Kollegen (vgl. [Han06]). Warum das so ist und was es für Entwicklerkarrieren bedeutet, darum geht es schließlich im letzten Abschnitt.

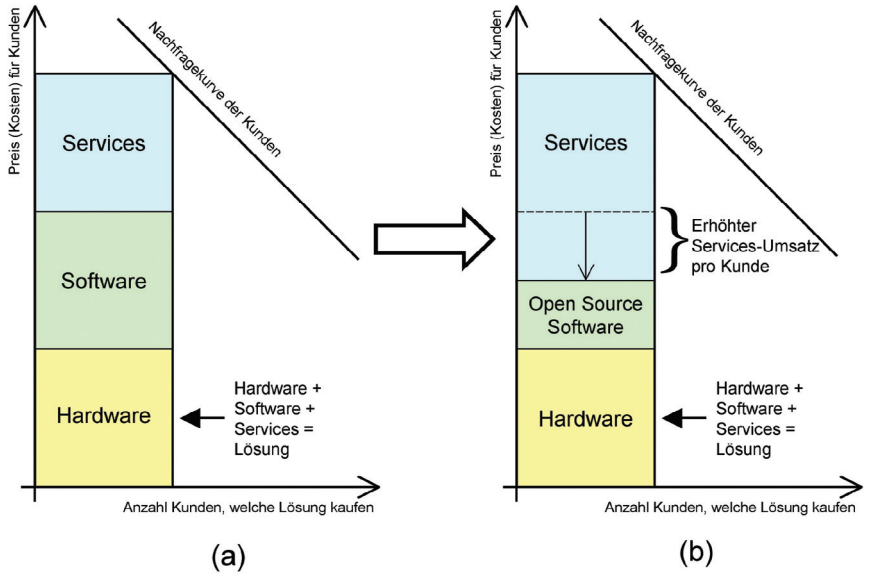

Abb. 1: Der IT-Stapel, bestehend aus Hardware, Software und Services

Die Sicht der Systemintegratoren

Systemintegratoren sind Firmen, die Kunden keine Softwareprodukte verkaufen, sondern *Lösungen*. Software ist hierbei lediglich Teil eines Pakets, das die Lösung für das Kundenproblem darstellt. Dadurch, dass eine Lösung immer mehrere Komponenten beinhaltet, sind Systemintegratoren in der Lage, die verschiedenen sich ergänzenden Komponenten gegeneinander auszuspielen. Deswegen sind die großen Integratoren, wie z. B. IBM, zu Open-Source-Software-Unterstützern geworden.

Betrachten wir dies zuerst an einem stark vereinfachten Beispiel: Sie verkaufen eine Lösung für den Handel. Ihre Lösung beinhaltet zwei Komponenten, das Betriebssystem und ihre Spezialsoftware. Dem Kunden ist Ihre Lösung 1.000 Euro wert. Davon müssen Sie 200 Euro an Microsoft abführen, weil Ihre Spezialsoftware Microsoft Windows als Betriebssystem benötigt. Somit verbleiben bei Ihnen nur 800 Euro. Viel besser wäre es, wenn Ihre Software mit Linux funktionieren würde, sodass bei angenommen 0 Euro Kosten für Linux die vollen 1.000 Euro bei Ihnen bleiben. Mit anderen Worten: Jeder Softwarehersteller wünscht sich, dass bis auf die eigene Software alle andere Software kostenlos wäre, sodass er den maximalen Umsatz abschöpfen kann, den ein Kunde bereit ist, für die Lösung zu zahlen.

Dieses Gedankenexperiment funktioniert nicht nur für zwei komplementäre Komponenten, die zu einer Lösung zusammengefügt werden, sondern auch mit mehreren. Große Systemintegratoren wie IBM decken den gesamten IT-Stapel (*IT Stack*) ab. Lösungen, die IBM seinen Kunden anbietet, bestehen aus Hardware, Software und Services. Je mehr Komponenten IBM extern – zum Beispiel von Microsoft – einkaufen muss, um so geringer ist IBMs Gewinn. Das bedeutet, dass IBM ein großes Interesse daran hat, eigene Komponenten einzusetzen oder – wenn dies nicht möglich ist – eine möglichst preiswerte Alternative zu verwenden, also etwa Linux statt Windows oder OpenOffice statt Microsoft Office. Dadurch, dass Kunden für das Gesamtpaket zahlen, statt für einzelne Komponenten, kann IBM mehr Gewinn abschöpfen.

„Oh," sagt der smarte IBM-Vertriebsmensch, „wenn Sie gleich fünf Jahre Services mitbestellen, bekommen Sie den Großrechner umsonst." Wenn der Vertrieb vernünftig aufgesetzt ist und nur das Gesamtergebnis zählt, ist es für den Vertriebsmenschen egal, woher der eigentliche Gewinn kommt und ob bestimmte Komponenten in einer Lösung offiziell für 0 Euro abgegeben werden. Um diese Preisgestaltungsflexibilität zu gewinnen, um somit Konkurrenten leichter aus dem Feld schlagen zu können und unliebsame Konkurrenten in den für IBM entstehenden Kosten zu reduzieren, investiert IBM heftig in Open-Source-Software bei Softwarekomponenten, in denen sie nicht den Markt dominiert. Abbildung 1 illustriert diese Verhältnisse etwas formaler. Das Bild zeigt, was passiert, wenn ein Systemintegrator in der Lage ist, eine zuvor mit Kosten verbundene proprietäre Software einer anderen Firma durch Open-Source-Software zu ersetzen: Bei gleichem Preis für den Kunden erzielt der Systemintegrator einen höheren Gewinn, weil mehr von dem Umsatz bei ihm verbleibt.

Kunden werden natürlich versuchen, in Verhandlungen den Gesamtpreis zu reduzieren. Wie aber oben bereits illustriert, wird der smarte Vertriebsmensch dies durch das Schnüren immer neuer und komplexerer, aber für den Kunden wertvollerer Lösungen kontern.

Es gibt weitere gute Argumente dafür, warum große Systemintegratoren Open-Source-Software unterstützen. Zum Beispiel können Sie den mit ihren Lösungen adressierbaren Markt erweitern, also potenziell an mehr Kunden verkaufen. Hier aber soll bereits diese eine Erklärung genügen, warum IBM und andere Systemintegratoren Open-Source-Software unterstützen.

Die Sicht der Softwarefirmen

Was bedeutet dies nun für unabhängige Softwarefirmen, die im Wesentlichen ein Produkt vertreiben? Sollen sie hoffen, dass die großen Systemintegratoren sie übersehen, sodass sie ein erfolgreiches Nischendasein führen können? Das mag in manchen Bereichen funktionieren, in vielen aber tut es dies nicht. Viele proprietäre Softwarekomponenten (Web-Server, Datenbanken, Benutzungsschnittstellen-Frameworks), die häufig gebraucht werden, werden schon lange von Open-Source-Software belagert oder sind von dieser bereits verdrängt worden.

Die Antwort ist bekannt: Statt proprietäre Software zu erstellen, haben sich viele Firmen gewandelt oder wurden neu gegründet, um als Service-Firmen Dienstleistungen rund um eine Open-Source-Software anzubieten. Dazu entwickeln diese Firmen an der Open-Source-Software mit, um die Expertise im Haus zu haben und die Open-Source-Software gut und erfolgreich zu machen. Die angebotenen Dienstleistungen umfassen häufig den Support der Software, Implementierungsarbeiten beim Kunden, das Hosting der Anwendungen über das Internet oder auch den Schutz der Kunden vor Klagen – sollte

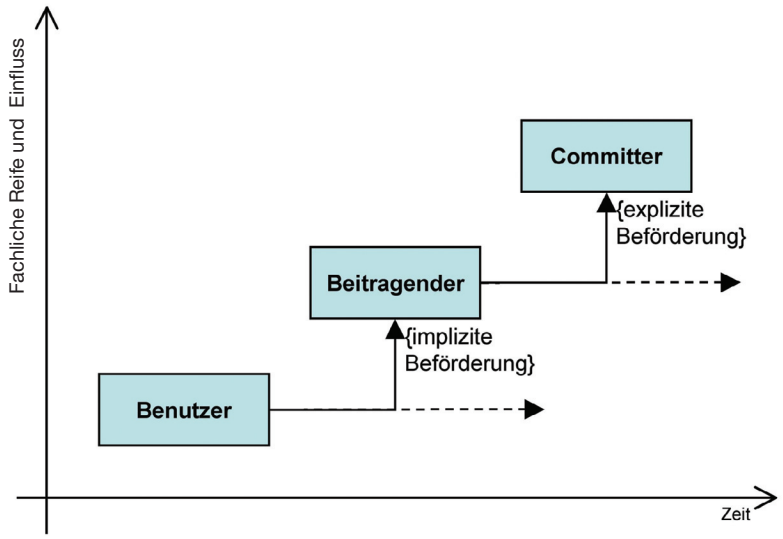

Abb. 2: Der Karrierepfad eines Open-Source-Softwareentwicklers

es rechtliche Probleme geben, etwa weil möglicherweise Rechte dritter Parteien verletzt wurden (die so genannte „Indemnification").

Warum machen Firmen dies? Mit entsprechend fachlich qualifizierten Entwicklern müsste es doch möglich sein, eine wettbewerbsfähige proprietäre Software zu entwickeln. Eine Antwort liegt in den aus Sicht der an einer Gemeinschaft beteiligten Firma geringeren Kosten von Open-Source-Software. Wenn viele Firmen zu Open-Source-Software beitragen, sinken die eigenen Kosten. Zwar sinkt damit auch der prozentuale Anteil am Kuchen, aber wenn dieser etwa durch Marktdominanz in absoluten Zahlen groß genug ist, lässt sich trefflich damit leben (besser einen kleinen Anteil an einem großen Kuchen als einen großen Anteil an einem kleinen Kuchen).

Machen wir aber folgendes Gedankenexperiment. Eine Firma beginnt, eine proprietäre Software für einen neuen Markt zu entwickeln. Ist der Markt interessant genug, werden bald weitere Startups folgen und Konkurrenzprodukte entwickeln. Schnell wird einer der weniger erfolgreichen Konkurrenten feststellen, dass er am Markt verlieren wird. Also besinnt er sich auf eine Open-Source-Strategie und legt seine Software offen, mit der Hoffnung, eine Gemeinschaft um diese Software zu bauen und den Markt doch noch zu erobern. „Besser mit Open-Source gewinnen als proprietär verlieren", lautet das Motto. Bei neuen Märkten kann sich heutzutage ein proprietärer Marktführer ausrechnen, dass diese Strategie gegen ihn angewendet werden wird. Also bleibt häufig nicht viel anderes, als die Software von vornherein offen zu legen.

Es gibt viele Variationen dieses Themas. Insbesondere ist

- „kommerziell entwickelte" Open-Source-Software von
- „gemeinschaftlich entwickelter" Open-Source-Software

zu unterscheiden. Bei kommerziell entwickelter Open-Source-Software behält eine einzelne Firma alle Rechte, während bei gemeinschaftlich entwickelter Open-Source-Software die Rechte bei einer freien Gemeinschaft liegen und durch die Lizenz geregelt werden. Ein Beispiel für die erste Kategorie ist „MySQL", ein Beispiel für die zweite ist „PostgreSQL". Kommerziell entwickelte Open-Source-Software ist natürlich bei Startup-Gründern und Risikokapitalgebern sehr beliebt, da sie sich höhere Gewinne versprechen. Auf Dauer halte ich ihren Erfolg aber für fraglich: Warum sollte ein Softwareentwickler zu einer Software beitragen, die jemand anderem gehört und wo er jegliche Rechte an seinem Beitrag verliert?

Die Sicht der Softwareentwickler

Eine Firma, die Dienstleistungen rund um eine Open-Source-Software anbietet, lebt und stirbt mit ihrer Expertise an der Software. Will man Kunden-Support und weitere Dienste glaubwürdig verkaufen, muss man diese Expertise demonstrieren. Die beste Art und Weise ist, einen oder mehrere *Committer* des Open-Source-Projekts zu beschäftigen. Um das zu verstehen, müssen wir erst einmal einen Schritt zurück treten und die Struktur von Open-Source-Projekten sowie die Rolle von Softwareentwicklern betrachten.

Ein typisches Open-Source-Projekt hat einen oder mehrere *Committer*, um die sich eine größere Gemeinde von Beitragenden schart, um welche herum sich noch einmal eine größere Gemeinde von Benutzern versammelt. Ein Entwickler kann potenziell jede dieser drei Rollen – Benutzer, Beitragender oder *Committer* – einnehmen. Abbildung 2 illustriert eine typische Karriere, bei der ein Entwickler von einem Benutzer zu einem *Committer* avanciert.

Die drei Rollen haben die folgende Bedeutung:

- *Benutzer:* Entwickler, die die Software einsetzen.
- *Beitragende:* Entwickler, die aktiv Feedback und *Patches* beitragen.
- *Committer:* Entwickler, die das Recht haben, *Patches* einzuspielen.

Ein typischer Arbeitsprozess sieht so aus: Ein Benutzer entdeckt einen Fehler und meldet ihn. Der Benutzer selbst oder jemand anderes (in der Rolle eines Beitragenden) analysiert den Fehler und löst das Problem. Der resultierende veränderte oder erweiterte Quelltext (*Patch*) wird auf eine Mailing-Liste oder an einen *Committer* geschickt, der ihn überprüft und bei Gefallen „committed", also in das zum Projekt gehörige Quelltext-Repository einspielt. Man erkennt die Bedeutung des *Committers*. Er überwacht die Integrität des Projekts; ohne seinen Willen wird kein Quelltext in das Projekt übernommen.

Softwareentwicker spielen vielfach mehrere Rollen. So ist ein Beitragender fast immer auch ein Benutzer. Und ein *Committer* ist fast immer auch ein Beitragender. In der Tat erwartet man von *Committern*, dass sie aktiv an der Programmierung beteiligt sind, was auch Sinn macht, weil dies ihre Expertise aufbaut und erhält.

Die Rolle des *Committers* ist aber vielschichtig. Er (oder sie) programmiert nicht

nur und überprüft Beiträge von anderen, ein *Committer* bestimmt Kraft seiner Position auch, in welche Richtung ein Projekt geht. Er kommuniziert zumeist aktiv diese Richtung und hilft dabei, Probleme zu lösen oder zumindest zu erklären. Dank seiner Reputation kann er die beteiligten Beitragenden in eine bestimmte Richtung lenken und Probleme schneller lösen. Gegenüber Benutzern hat ein *Committer* eine hohe Sichtbarkeit.

Dank dieser Rolle ist ein *Committer* für eine Service-Firma wichtig. Insbesondere die hohe Sichtbarkeit eines *Committers* in der Nutzergemeinde schafft Vertrauen, dass sein Arbeitgeber – d. h. die Service-Firma – ihre Dienstleistung auch erbringen kann. Für den *Committer* ist diese Position somit bares Geld wert. Eine Studie hat ergeben, dass *Committer* an Projekten der Apache Software Foundation mehr Geld verdienen als gleich qualifizierte Softwareentwickler, die keine solche herausgehobene Position innehaben (vgl. [Han06]). Es ist anzunehmen, dass sich diese Ergebnisse auch auf andere Projekte übertragen lassen.

Es gäbe noch viel darüber zu sagen, wie ein Softwareentwickler eine Open-Source-Karriere am besten angeht, wie man am effektivsten vom Benutzer zum *Committer* befördert wird und wie dies die Verhandlungsposition und Loyalität gegenüber potenziellen Arbeitgebern verändert. Dies aber soll weiteren Artikeln vorbehalten bleiben. ■

Literatur

[FSF] Free Software Foundation, siehe: www.fsf.org

[Han06] I.-H. Hann, J. Roberts, S. Slaughter, R. Fielding, Economic Returns to Open-Source Participation: A Panel Data Analysis, nicht veröffentlichte Studie, Univ. of Southern California, 2006

[OSDL] Open Source Development Labs, siehe: www.osdl.org