Wer zahlts?

Geschäftsmodelle mit Open-Source-Software

Bild: Albert Hulm

Open-Source-Software ist schon lange aus der Ecke der hobbymäßigen Freizeitprogrammierer herausgekommen – wenn sie da je war. Trotzdem hält sich das Bild von kostenloser Software, die zwar nicht an „kommerzielle“ Produkte heranreicht, aber immerhin gratis ist. Wir zeigen, wie falsch dieses Bild ist.

Von Sylvester Tremmel

Die Entwicklung von Software kostet Zeit und Geld, Aufwendungen, die irgendwie getragen werden müssen. Ob man für ein Produkt bezahlen muss, hat wenig damit zu tun, ob es Open-Source-Software ist, proprietäre Freeware gibt es genauso wie teure Open-Source-Produkte. Auch welche Produkte millionenschwere Marktführer sind und welche eine kleine Hobbybastelei, lässt sich nicht an der Lizenz erkennen.

Wir geben im Folgenden einen Überblick darüber, woher das Geld für Open-Source-Entwicklung kommt und welche verschiedenen Finanzierungsmodelle es dafür gibt. Doch zunächst gilt es, einige Missverständnisse und Unklarheiten zu beseitigen, die mit dafür verantwortlich sind, dass sich das genannte schiefe Bild so hartnäckig hält.

FOSS und FLOSS

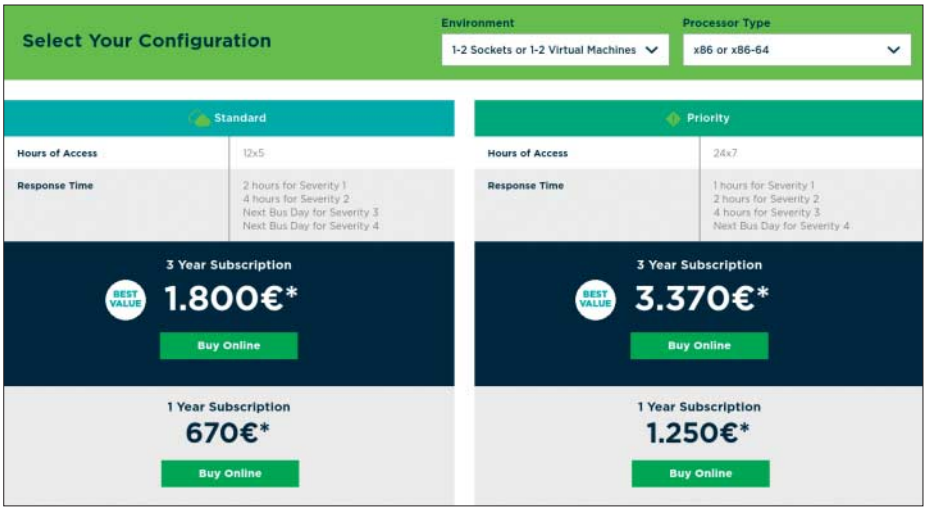
Der Ausdruck Open-Source-Software wird gelegentlich genutzt, um Software zu bezeichnen, deren Quellcode zugänglich ist. Das ist naheliegend, weil „open source“ eben „offener Quellcode“ bedeutet, aber meistens ist mit „offen“ mehr gemeint als bloß „lesbar“. Sehr grob gesagt soll der Code eines Produktes – zumindest für dessen Käufer – auch veränderbar sein. Es geht dabei sowohl um das Recht als auch die praktische Möglichkeit, den Code zu verändern. Unter anderem will man sicherstellen, dass das Produkt dauerhaft und beliebig genutzt und auch verbessert werden kann. Open-Source-Software ist also Software, die beliebig eingesetzt werden darf, deren Quellcode zugänglich ist und deren Quellcode verändert werden darf.

Zur Unterscheidung spricht man auch von „source available“ oder „shared source“, wenn Quellcode einsehbar ist, aber nicht frei verändert werden darf. Diese Unterscheidung wird aber, auch aus Gründen des Marketings, nicht immer exakt eingehalten. Hinzu kommt die „offizielle“ Definition von Open Source, die die Open Source Initiative (OSI) verwaltet, siehe ct.de/ysp1. Die Definition der OSI schließt die obigen Punkte mit ein, spezifiziert aber noch anderes, etwa Verbote, Personengruppen oder Einsatzzwecke der Software zu diskriminieren.

Daneben wird auch von „freier Software“ gesprochen, eine deutlich politischere Wortwahl: Befürworter von „freier Software“ – allen voran die Free Software Foundation (FSF) – sehen unfreie Software als gesellschaftliches Problem an. Ihrer Meinung nach ist der Ausbau von freier Software moralisch geboten und muss aus gesellschaftlichen Überlegungen heraus vorangetrieben werden. Fürsprecher von „open source“ – hier ist die OSI maßgeblich – sehen mehr praktische Aspekte im Vordergrund und befürworten Open Source, weil sich damit bessere Programme schreiben ließen. Politische oder ideologische Implikationen wollen sie eher ausblenden, was die FSF ablehnt, siehe ct.de/ysp1. So unterschiedlich die Philosophien sind: Freie Software ist fast notwendigerweise auch Open-Source-Software im Sinne der OSI, und Open-Source-Software ist in den meisten Fällen auch frei nach den Regeln der FSF. Im Englischen wird solche Software daher häufig auch als „free and open source software“ bezeichnet – FOSS.

Leider ist auch der Begriff „frei“ Ausgangspunkt für ein übliches Missverständnis: „Frei“ kann als Synonym zu „gratis“ verstanden werden, was besonders im Englischen weit verbreitet ist. Freie Software muss aber keineswegs kostenlos sein. Trotzdem hält sich das Missverständnis hartnäckig, obwohl seit Jahren dagegen angeschrieben wird, unter anderem mit dem Slogan „frei wie in Redefreiheit, nicht wie in Freibier“.

Als Reaktion auf die Doppelbedeutung von „frei“ wurde der Begriff „FLOSS“ geprägt, „free/libre open source software“, weil dem lateinischstämmigen „libre“ nicht die Konnotation „kostenlos“ anhaftet. Letztendlich haben diese Missverständnisse und die Versuche, dagegen immune Begrifflichkeiten zu finden, aber nur zu einer relativ unübersichtlichen Gemengelage geführt. Man muss darauf achten, wer einen Begriff nutzt, um ihn einordnen zu können. Wir werden in diesem Artikel schlicht von „Open Source“ reden, obwohl die Softwareprojekte, um die es geht, auch „frei“ im Sinne der FSF sind.

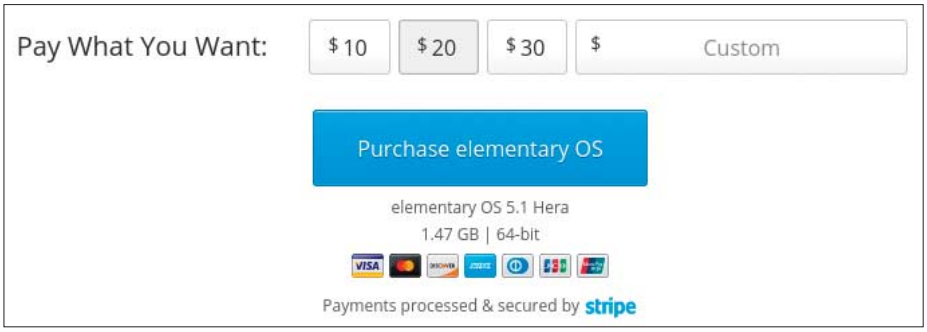


Linux-Distributionen wie die von SUSE sind keineswegs grundsätzlich kostenlos. Open Source sind sie trotzdem.

Freibier

Aber wenn der Quellcode einsehbar ist und verändert werden darf, bedeutet das nicht, dass die Software faktisch gratis ist; wer kauft das schon, wenn man es auch kostenlos bekommen kann? Dieser Frage liegt eine falsche Prämisse zugrunde: Weder Produkt noch Quellcode kann man unbedingt kostenlos bekommen.

Wie gesagt, Open-Source-Software bedeutet, dass der Quellcode verändert werden kann und darf. Der Quellcode muss aber nicht öffentlich oder kostenlos verfügbar gemacht werden. Nur Empfänger des Produkts müssen Zugang bekommen, zum Beispiel indem der Code mit dem Produkt ausgeliefert wird. Solche Empfänger können durchaus zahlende Kunden sein. Erst recht nicht erforderlich ist es, eine kompilierte ausführbare Anwendung kostenlos anzubieten, damit sie Open Source wird.



Zahlen, was man will. Bei „Custom“ kann man auch null Dollar eintragen und bekommt elementary OS dann kostenlos.

Produkte verkaufen

Daher ist es auch im Open-Source-Umfeld durchaus verbreitet, von der Softwareentwicklung zu leben, indem man schlicht die entwickelte Software verkauft. Gerade bei größeren Softwareprojekten nimmt der Aufwand des Kompilierens aus dem Quellcode schnell immense Ausmaße an, insbesondere weil ja auch jedes Update und jeder Patch kompiliert und eingepflegt werden will. Folgerichtig verfolgen auch eher größere Softwareprojekte das Geschäftsmodell, ihr Produkt einfach direkt an (End-)Kunden zu verkaufen. Bekannte Beispiele sind Linux-Distributionen, besonders solche die sich an Firmen richten, wie **Red Hat Enterprise Linux** oder **SUSE Linux Enterprise Server**. Auch vom verbreiteten **LibreOffice** gibt es kostenpflichtige Versionen, die sich ebenso eher an Firmen richten (siehe S. 74).

In diesem Business-to-Business-Bereich sind Kunden auch vergleichsweise willens zu zahlen, weil es selten kostenlose Alternativen gibt. Schwieriger haben es zum Beispiel Linux-Distributionen, die sich an Privatanbieter richten. Dort gibt es nicht nur viele kostenlose Angebote, die große Mehrheit aller Rechner wird ohnehin schon mit einem Betriebssystem gekauft: Microsoft Windows oder macOS. Die sind zwar nicht gratis, aber beim Kauf eines PCs sind die Kosten des Betriebssystems in der Regel schon fest mit eingepreist. Welcher Teil des Kaufpreises auf die Nutzungslizenz des Betriebssystems entfällt, ist oft intransparent und diese Kosten werden nicht als separater Preis für ein separates Produkt wahrgenommen.

Das behindert zwar alle alternativen Betriebssysteme, aber zumindest auf PCs sind die auch fast sämtlich Open Source. Nichtsdestotrotz kann man auch Privatanwendern Open-Source-Betriebssysteme verkaufen: Zum Beispiel nutzt die Linux-Distribution **elementary OS** ein „Zahle, was Du willst“-Modell. Kunden kaufen das Produkt und können den Kaufpreis selbst wählen. Vorgeschlagen werden 10, 20 oder 30 Dollar, aber man darf auch 0 Dollar als Kaufpreis eingeben. Tatsächlich macht das die große Mehrheit der Kunden, aber der Rest generiert genügend Einnahmen, um die Distribution zu finanzieren. Im Prinzip ähnelt dieses Verkaufsmodell einer dauerhaft laufenden Werbeaktion, die manchen Kundenkreisen das Produkt schenkt – nur, dass die Kunden sich selber den Kreisen „zahlungswillig“ und „-unwillig“ zuordnen.

Ähnlich ist das Geschäftsmodell, verschiedene Vertriebskanäle zu nutzen und dasselbe Produkt auf dem einen kostenpflichtig und auf dem anderen kostenlos anzubieten. Was kurios anmutet, kommt insbesondere bei App-Stores vor: Die Malsoftware **Krita** gibt es etwa kostenpflichtig über Steam und Microsofts Store, obwohl der Hersteller auch kostenlose Installer auf seiner Homepage anbietet. Der Artikel auf Seite 72 beleuchtet das Konzept näher.

Dunkle Wolken

Im vergangenen Jahr haben einige Anbieter von Datenbank-Systemen von sich reden gemacht, weil sie ihre vormals als Open Source angebotenen Produkte ganz oder teilweise auf proprietäre Lizenzen umstellten. Bekannte Beispiele sind **Redis** – wobei hier nur manche Module und nicht das Kernprodukt betroffen sind –, **CockroachDB** und **MongoDB**.

Das Problem ist, dass große Cloud-Anbieter wie Amazon die Open-Source-Produkte dieser Hersteller übernahmen und den eigenen Kunden „as a service“ zur Verfügung stellten. Solche Software-as-a-Service-Angebote sind aber die Strategie, mit der sich Redis & Co. selbst finanzieren. Die Cloud-Anbieter konkurrieren daher nicht nur direkt mit diesen Firmen, sie können – dank ausgefeilter Integration mit den eigenen Cloud-Produkten – vielen Kunden auch ein besseres Produkt bieten.

Mit der Relizenzierung ihrer Produkte verbieten die Datenbankhersteller in der Regel nur genau dieses Verhalten. Das reicht aber, um die Open-Source-Definition der OSI zu verletzen. Der Softwareempfänger, zum Beispiel Amazon, hat eben nicht mehr das Recht, die Software beliebig einzusetzen. Die Hersteller argumentieren, dass sie damit ein Problem der Open-Source-Definition beheben. Gegner vertreten den Standpunkt, dass solche Lizenzen zu recht nicht als Open Source gelten und das Problem eher die Marktmacht der Cloud-Anbieter ist – ganz unabhängig vom Lizenzierungsmodell.

Milde Gaben

Von solchen Pay-what-you-want-Modellen ist es nur noch ein kleiner Schritt zu Finanzierungsmodellen, die komplett auf freiwilligen Spenden basieren. Das wird häufig mit dem klischeehaften, einzelnen Programmierer im Keller assoziiert, der davon nicht einmal leben kann. Auch dies ist ein Zerrbild. Zwar gibt es viele, gerade kleinere Projekte, die maßgeblich von Enthusiasten und deren Freizeit getragen werden. Das bedeutet aber nicht, dass spendenbasierte Finanzierungsmodelle nicht skalieren könnten oder langfristig untragbar wären. Ein bekanntes Beispiel für spendenfinanzierte Großprojekte ist die **Wikipedia** mit ihren diversen Schwesterprojekten. Getragen werden diese von der gut 100 Millionen Dollar schweren Wikimedia Foundation, die sich maßgeblich aus Spenden finanziert – im Durchschnitt 15 Dollar pro Spende.

Das 3D-Modellierungs und -Animationsprogramm **Blender** ist ein anderes Beispiel. Das in der Medienbranche weit verbreitete Programm wird maßgeblich über den spendenfinanzierten Blender Development Fund finanziert. In den zahlen neben Privatpersonen auch industrielle Großspender ein. Die fördern so ein Programm, das sie in ihrem Workflow nutzen oder von dem sie anderweitig profitieren. Monatlich kommen knapp 100.000 Euro zusammen, was ausreicht, um unter anderem 15 Entwickler zu beschäftigen.

Auch kleineren Projekten stehen solche Finanzierungsmodelle offen. Statt mit oft geringer Reichweite selbst Spenden einzusammeln oder Sponsoren anzuwerben, können sie sich ähnlich wie Blender oder die Wikipedia von einem Fonds finanzieren lassen. Die haben in der Regel eine größere Reichweite und bieten mehr Planungssicherheit für die finanzierten Projekte. Ein Beispiel aus Deutschland ist der **Prototype Fund**, ein von der Open Knowledge Foundation Deutschland betreutes Förderprogramm, das Softwareprojekte im öffentlichen Interesse unterstützt, siehe ct.de/ysp1. Finanziert wird der Prototype Fund vom Bundesministerium für Bildung und Forschung.

Gemeinschaftssinn

Mit Ausnahme des Prototype Funds können alle bisher genannten Finanzierungsmodelle auch von Closed-Source-Projekten genutzt werden. Warum sind Projekte also Open Source? Ein Grund ist, dass sich mit Open-Source-Software wesentlich leichter und besser eine Community aufbauen lässt. Der Umstand, dass jeder selbst Verbesserungen am Code vornehmen kann, zieht häufig Personen an, die sich dafür interessieren, ermutigt reine Anwender dazu, sich am Projekt zu beteiligen, und erlaubt Dritten, das Projekt auf die Arten zu verbessern, die ihnen wichtig sind. Eine so wachsende Anwendergemeinschaft ist für Projekte grundsätzlich förderlich, egal wie sie sich finanzieren.

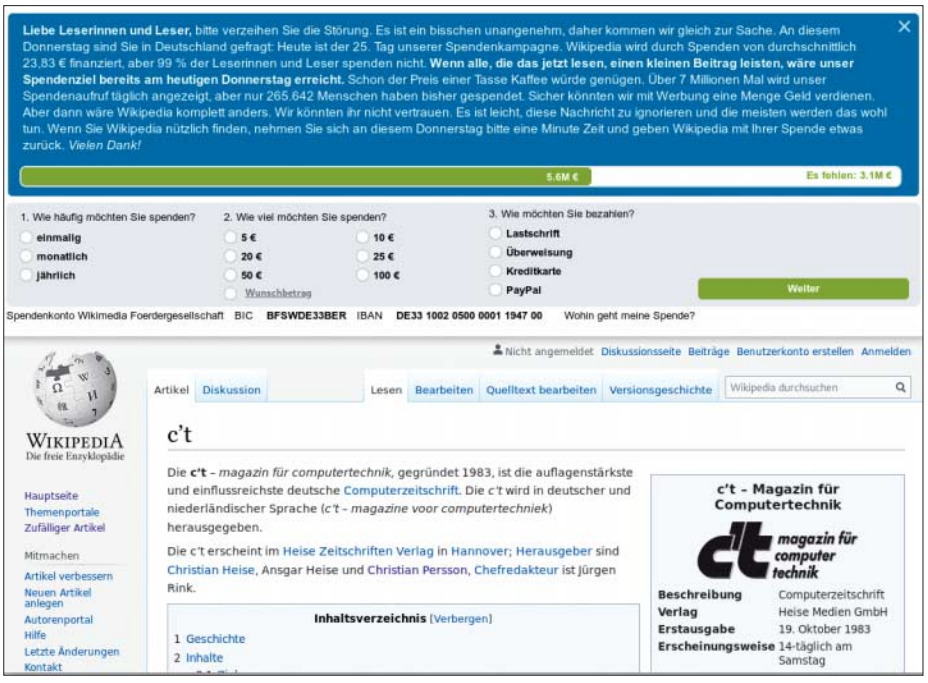
Vor allem aber gibt Open Source der Gemeinschaft gewisse Garantien. Zum Beispiel kann eine einmal mit der Community im Quellcode geteilte Anwendung dieser Gemeinschaft nicht mehr einfach entzogen werden. Ein Anbieter proprietärer Software kann seine Software kostenlos weitergeben, um eine Nutzer-Community aufzubauen und dann über Nacht anfangen, Geld dafür zu verlangen. Anwender wären gezwungen zu zahlen oder würden zumindest von der weiteren Entwicklung abgeschnitten. Der Community um eine Open-Source-Software kann dergleichen nicht passieren: Zumindest von der letzten als Open Source veröffentlichten Version steht der Quellcode zur Verfügung und darauf aufbauend kann die Software unabhängig vom ursprünglichen Hersteller weiterentwickelt werden. Zudem erzwingen Open-Source-Lizenzen oft, dass auch zukünftige Versionen der Software der Gemeinschaft im Quellcode zur Verfügung stehen. Details zu diesem Copyleft genannten Konzept erläutert der Artikel auf Seite 68.

Solche Sicherheiten machen die Community für alle Teilnehmer attraktiver, weil diese wissen, dass ihre Arbeit nicht irgendwann einseitig monetarisiert oder gänzlich nutzlos wird. Das ist auch einer der Gründe, warum Organisationen, die auf das Gemeinschaftswohl ausgerichtet sind, oft Open Source bevorzugen, fördern oder erzwingen. Auch der erwähnte Prototype Fund fördert deshalb nur Open-Source-Projekte.

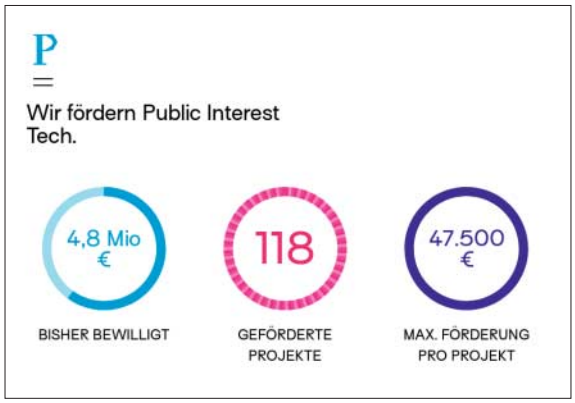
Konkurrenz ist gut fürs Geschäft

Außerdem machen solche Sicherheiten die Community für andere kommerzielle Anbieter attraktiv: Drittfirmen können einerseits im horizontalen Markt um die Software Dienstleistungen und Produkte anbieten, etwa Add-ons oder Support. Und sie können eigene Produkte entwickeln und verkaufen, die auf der Software aufbauen, der sogenannte vertikale Markt. Beides wäre mit erheblichem unternehmerischen Risiko verbunden, wenn Open Source nicht die dauerhafte Verfügbarkeit des Kernproduktes garantieren würde.

Viele Softwareprojekte nehmen dieses Prinzip auch zum Geschäftsmodell: Ein für sich allein genommen schon funk-



Kennen die meisten: Wikipedia bittet um Spenden. Auch solche gigantischen Projekte lassen sich über freiwillige Gaben langfristig finanzieren.



Der mit öffentlichen Geldern finanzierte Prototype Fund fördert Software-Projekte mit Nutzen für die Gesellschaft. Damit die dauerhaften Zugriff hat, müssen die Projekte Open Source sein.

tionaler Kern wird als – oft kostenlose – Open-Source-Software entwickelt, Geld wird mit darum herum angebotenen Dienstleistungen und – eventuell proprietären – Add-ons verdient. Das im Artikel auf Seite 70 beschriebene **VCV Rack** ist ein Beispiel einer solchen Entwicklung. Auch viele der eingangs genannten Linux-Distributionen haben kostenlose Pendants, die auch für sich allein genommen einsetzbar sind, und die einen Großteil der Community binden, etwa **Fedora Linux** für Red Hat oder **openSUSE**.

Durch den offenen Kern können zwar auch Mitbewerber zu den eigenen kommerziellen Angeboten in Konkurrenz treten, aber häufig gilt hier: Konkurrenz belebt das Geschäft. Der offene Kern und die von Dritten eingebrachten Verbesserungen befeuern die Community so stark, dass deren Wachstum die negativen Effekte zusätzlicher Konkurrenz überwiegt. Eine Ausnahme beschreibt der Kasten auf Seite 64.

Unter Umständen kann die Verwaltung des Kerns sogar komplett in die Hände einer neutralen Instanz gegeben werden, wie das etwa beim **Linux-Kernel** der Fall ist, der unter der Obhut der Linux Foundation steht. Auch das **LibreOffice**-Projekt wird so verwaltet. Der Artikel auf Seite 74 beschreibt das System genauer.

Die Monetarisierung über die Community kann auch noch deutlich indirekter ausfallen: Die Mozilla-Stiftung, die unter anderem die Entwicklung des Browsers **Firefox** maßgeblich finanziert, zieht einen Großteil ihres Einkommens aus Verträgen mit Suchmaschinen. Diese zahlen dafür, dass sie die voreingestellte Suche in Firefox sind. Attraktiv für die Suchmaschinen – und ertragreich für Mozilla – ist das nur, solange Firefox eine große Gemeinschaft von Anwendern aufweist.

Noch indirekter ist der Nutzen bei Projekten wie Googles Chrome-Browser und dem Android-Betriebssystem. Beide haben als Kern ein Open-Source-Projekt: den Browser **Chromium** beziehungsweise das **Android Open Source Project** (AOSP). Aber auch die darauf aufbauenden Projekte sind für Endkunden kostenlos erhältlich und befördern – zusammen mit ihren Open-Source-Kernen – große Communitys aus Anwendern und Entwicklern. Chrome und vor allem seine Browser-Engine Blink dominieren den Browser-Markt ähnlich stark wie Android den Markt der mobilen Betriebssysteme.

Für Google rechnet sich das schon deswegen, weil die Firma so Einfluss auf große Teile der für sie relevanten IT-Infrastruktur hat. Sie kann darüber auch die Verbreitung und Nutzung eigener Produkte befördern, etwa die Google-Suche im Browser oder die eigenen Apps auf dem Smartphone. Zusätzlich profitiert das Unternehmen über Angebote wie den Play-Store. Dritte, die das geschaffene Ökosystem für eigene Angebote nutzen, zahlen Gebühren an Google für die Abwicklung von Verkäufen.

Fazit

Open-Source-Software monetarisiert sich also letztlich immer über ihre Community. Entweder, weil Anwender schlicht dafür bezahlen müssen, oder weil die Community andere Einnahmequellen bietet. Angefangen bei direkten Spenden oder Förderungen, zu kostenpflichtigen Zusatzangeboten, die die Open-Source-Entwicklung querfinanzieren. Viele Projekte nutzen auch Mischfinanzierungen; man kann beispielsweise an viele Projekte spenden, auch wenn sie andere primäre Einnahmequellen haben. In jedem Fall ist die Größe und Dynamik der oft globalen Communitys ein entscheidender Faktor für den Erfolg von Software – und Open Source fördert solche Gemeinschaften massiv. *(syt@ct.de)* c't

Weitere Infos: *ct.de/ysp1*

Branchengrößen vereint: Für die über tausend Unternehmensmitglieder der Linux Foundation ist es wirtschaftlich sinnvoll, gemeinsam Linux zu fördern.