Aus Linux-Magazin 05/2011

Das Single-Vendor-Konzept als erfolgversprechendes Modell für OSS-Unternehmen

Fest vernetzt

Dirk Riehle , Markus Feilner

Mit welchem Geschäftsmodell verdienen Open-Source-Softwareschmieden Geld? Der bislang einzige deutsche Open-Source-Professor gibt einen Überblick über Erfolgsfaktoren und zeigt die richtige Community-Strategie. Eine wichtige Rolle spielt dabei das Single-Vendor-Modell.

Wer als Unternehmen mit Open-Source-Software verdienen will, braucht eine starke Community. Die zu organisieren ist oft keine triviale Aufgabe. Dennoch: Gerade in den letzten Jahren zeigen immer mehr Firmen, wie das gehen kann. Meist folgen die dem Geschäftsmodell des "Single vendor commercial open source"-Ansatzes, bei dem ein Unternehmen als einziger Anbieter einer Open-Source-Software alle Fäden in der Hand hält.

Über die Hälfte

Mit einer engagierten Community lässt sich viel gewinnen – wenn die Firma die Grundbegriffe der freien Softwarewelt versteht und dieses Wissen klug einsetzt. MySQL, Sugar CRM, Jaspersoft, Alfresco – die Liste erfolgreicher Unternehmen, die dieses Modell erfolgreich umsetzen, ließe sich noch verlängern.

Glaubt man Gartner [1], stammt spätestens 2012 bereits mehr als die Hälfte der mit OSS erzielten Gewinne aus Firmen, die als einziger Hersteller einer freien Software im Zentrum des jeweiligen Projekts stehen.

Theorie

Die wirtschaftswissenschaftlichen Hintergründe des Single-Vendor-Modells sind bisher nur teilweise erforscht, erste Erkenntnisse brachten die Untersuchungen von Brian Fitzgerald [2] oder das Flossmetrics-Projekt der EU (Free/Libre and Open Source Software Metrics). Es untersuchte in einer Studie 120 Firmen, die den Großteil ihrer Einnahmen aus OSS-Produkten beziehen.

Flossmetrics teilt die Unternehmen in sechs relevante Kategorien ein [3]: Die Spanne reicht von reinem Consulting über Plattform-Provider wie Suse oder Red Hat, Produktspezialisten à la Alfresco bis zu Badgeware wie Open Bravo oder Open EMM. Dazu kommen Firmen, die ihre Produkte unter verschiedenen Lizenzmodellen vertreiben (Split Releases oder Twin Licence).

Es bietet sich an, Open-Source-Projekte in kommerzielle und Community-Projekte zu unterteilen, wobei letztere den weitaus größeren Teil ausmachen. Während eine Gemeinschaft von Entwicklern Community Open Source betreibt, steht hinter kommerziellem Open Source, vor allem mit dem Single-Vendor-Modell, ein Interessenvertreter, der finanziellen Erfolg erreichen will.

Klassische Community-Projekte sind der Linux-Kernel, der Apache-Webserver oder die PostgreSQL-Datenbank. Deren komplette Software unterliegt der gleichen Lizenz, jeder darf sich der Programme in vollem Umfang bedienen und damit Geld verdienen. In zunehmendem Maße nehmen dabei Non-Profit-Stiftungen wie die Apache- oder Eclipse-Foundation die Interessenvertretung der Freiwilligen wahr und folgen bei ihren Aktionen meist auch den Vorgaben der Entwicklergemeinschaft.

Mit Community-Projekten lässt sich in der Regel auf dreierlei Art Gewinn erwirtschaften: Ein Softwarehaus mag Consulting und Support anbieten, kommerzielle Produkte basierend auf der freien Codebasis entwickeln oder OSS nutzen, um auf anderen Ebenen des Software-Stacks Einkünfte zu erwirtschaften [4], so wie das beispielsweise Provider tun.

Abbildung 1: Wer mit OSS Erfolg haben will, muss auf die Befindlichkeiten seiner Community hören.

Single Vendor

Anders die Single-Vendor-Firmen. Sie kontrollieren ein Open-Source-Projekt, das sie in der Regel auch selbst entwickelt haben. Das Unternehmen hält volles Copyright am Code, in der Regel festigen (Software-)Patente und eingetragene Marken diese zentrale Stellung.

Michael Olson, der in [5] unter anderem den Begriff Dual Licencing hinsichtlich geistiger Urheberschaft diskutiert hat, beschreibt diese volle Kontrolle als essenziell für ein funktionierendes Geschäftsmodell. In der Konsequenz tun sich solche Firmen erwartungsgemäß schwer mit Code-Contributions von außen. Um ihre dominante Stellung zu wahren, muss ein Patch fast zwangsläufig vollständig in das Copyright der Firma wechseln, auch wenn Larry Augustin dem widerspricht und glaubt, dass das Recht zur Relizenzierung vollständig reiche [6].

In jedem Fall unterscheiden sich Single-Vendor-OSS-Unternehmen von klassischen Softwareschmieden, schon weil sie ihre Produkte nicht nur als Binärpakete, sondern auch im Quelltext freigeben. Open-Source-Lizenzen führen zu Open-Source-Unternehmen.

Weil sie die Lizenzhoheit über ihre Quellen halten, steht es ihnen auch frei, jedem Kunden individuelle Freiheiten zu garantieren und die Software unter verschiedenen Lizenzen zu vertreiben. Typischerweise unterliegt die freie Open-Source-Variante der GPL, womit die Firma die Akzeptanz erhöhen und mögliche (proprietäre) Mitbewerber ausbremsen möchte. Parallel dazu gibt es kommerzielle Versionen mit klassischen Softwarelizenzen.

Einnahmequellen

Das Geschäftsmodell Single Vendor, also die Kombination aus Wertschöpfungsstrategien und unterstützenden Geschäftspraktiken und -funktionen baut laut Bearden [7] auf vier Kategorien, um Gewinn zu erzielen:

- Core-Produkte: Manche Kunden bezahlen für die Software, weil für sie Open-Source-Lizenzen nicht in Frage kommen, zum Beispiel weil sie dazu verpflichtet sind, zertifizierte Umgebungen zu betreiben, oder Software in Partnerprodukte einbetten müssen oder wollen.
- Whole-Produkte: Kunden bezahlen für Anwendungen, die auf der freien Softwarevariante aufbauen und so Funktionserweiterungen bieten.
- Operational Comfort: Manche Anwender lassen es sich einiges kosten, dass ein Dritter die Funktionalität

der Software sicherstellt. In diese Kategorie fallen Hotlines, Supportverträge und Subscriptions.
- Consulting Services: Training, Beratung, Dokumentation und Dienstleistungen rund um die Implementierung freier Software.

Aus dieser und ähnlichen Kategorisierungen entstanden in der Vergangenheit Bezeichnungen wie besagtes Dual-Licence-Modell. Das Wortspiel Freemium (Free und Premium, [8]) bedeutet nichts anderes als das gezielte Zurückhalten wertvoller Features für die Premium-Version einer freien Software. Von Lampitt [9] stammt der Begriff des Open-Core-Modells, das Freemium mit dualer Lizenz kombiniert.

Business-Funktionen und Organisationsstruktur

Ganz neue Ansätze sind dagegen im Bereich des Verkaufens und der Software-Entwicklung gefragt. Wer den Sourcecode seines Produkts veröffentlicht, hat die Chance, eine engagierte Community zu gründen, die vielerlei positive Einflüsse in die Firma zurückgeben kann. Dieser Input vermag Unternehmen sogar einen Vorteil gegenüber der Konkurrenz zu verschaffen.

Vor allem fünf Bereiche der Organisationsstruktur sind davon besonders betroffen: Neu einzuführen ist das jetzt benötigte Community-Management, der Vertrieb wird von der größeren Verbreitung und einzelnen, begeisterten Usern (Champions) ebenso profitieren, wie das Marketing an Transparenz und Glaubwürdigkeit gewinnen kann. Motivierte Anwender versorgen das Produktmanagement mit Ideen, die internen Entwickler freuen sich über schnellere und direktere Rückmeldungen aus der Community, die im Idealfall nebenbei auch den Support entlasten.

Auf der anderen Seite setzt sich eine Firma, die den Weg des Going GNU-Public beschreitet, auch Gefahren aus. Patentklagen der Konkurrenz oder der Verlust geistigen Eigentums sind dabei nur der Anfang. Im schlimmsten Fall endet das eigene Produkt im Laufe der Zeit in einer direkten Konkurrenzsituation mit der Open-Source-Variante. Das zu vermeiden erweist sich bisweilen als nicht triviale Aufgabe, wobei ein gutes Gespür für die Community unerlässlich ist.

Der Hersteller muss die Infrastruktur schaffen

Nicht zu vernachlässigen sind auch die Kosten, um einer Community die passende Infrastruktur zu bieten. Ohne engagierte Anwendergemeinschaft ist aber jedes freie Projekt zum Scheitern verurteilt. Sie umfasst sowohl User als auch Entwickler, wobei beim Single-Vendor-Modell ja die Firma selbst die Programmierung erledigt und Beiträge der Community nur integriert.

Neben einem Ökosystem an Partnern und Entwicklerfirmen, die zusätzlichen Mehrwert rund um das Produkt generieren, bedarf es einer sich im Idealfall selbst supportenden Userbase. Die Software frei verfügbar zu machen senkt die Hemmschwelle der Anwender. Mehr User werden sie ausprobieren, weil nicht zwangsläufig Kosten entstehen und sie die Gefahr eines Vendor-Lock-in eher gering einschätzen.

Der verbreitete Anwenderglaube, man könne im Notfall die Programme selbst weiterentwickeln, erweist sich allerdings meist eher als recht naive Betrachtungsweise. Realistischer ist die Einschätzung, dass Anwender bei einem kostenlosen Open-Source-Produkt eher die Möglichkeit haben und die Bereitschaft zeigen, Fehler oder Probleme selbst oder mit Hilfe der Community zu lösen. So kann der Hersteller die Verbreitung seiner Software fördern, ohne explodierende Supportkosten befürchten zu müssen – ein Problem, das konventionellen Entwicklern nur allzu vertraut ist.

Eine Community fördern

[10] und [11] beschreiben detailliert, wie Firmen Communities säen und aufbauen, inklusive der benötigten Infrastruktur wie Foren, Wikis, Mailinglisten und Softwareschmieden für den Quellcode, zum Beispiel mit Webportalen wie bei Source- oder Sugarforge.

Eine weitere Aufgabe ist der Aufbau sozialer Strukturen, zum Beispiel mit Belohnungen für besonders aktive Mitglieder. Auch das Marketing muss die Community im Blick behalten und kann dann beispielsweise Untergruppen mit ähnlichen Bedürfnissen identifizieren und ihnen passende Angebote unterbreiten. Community-Manager versuchen hier, Win-win-Situationen zu schaffen, die allen Seiten Vorteile bereiten

Larry Augustin hat in [12] den klassischen Weg skizziert, den ein User bei kommerzieller Open-Source-Software durchläuft, bis er zum Kunden wird (Abbildung 2). Dahinter steht ein neuartiges Modell der Lead-Erzeugung, das den klassischen Ansatz (Pre Sales to Sales) ersetzt: Anwender testen die Software, ohne mit dem Hersteller in Kontakt zu kommen. Der jedoch kann über freiwillige Registrierung oder Statistiken wertvolle Daten erheben und so seine Leads qualifizieren.

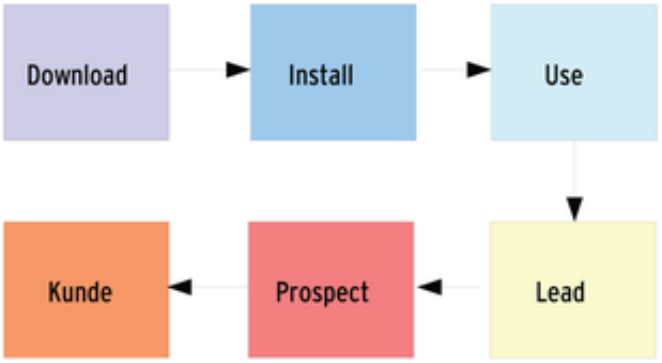

Abbildung 2: Der typische Sales-Funnel (Vertriebstrichter) eines kommerziellen Open-Source-Unternehmens. Im Gegensatz zum klassischen Vertrieb kommt hier der Kontakt meist vom Anwender selbst, nachdem dieser die Software ausgiebig getestet hat.

In den meisten Fällen jedoch wartet die Firma, bis ein (nicht zahlender) Anwender auf sie zukommt und nach Services aus dem kommerziellen Segment des Angebots fragt. Während im klassischen Ansatz der User die Firma hauptsächlich übers Marketingmaterial kennenlernt, hat der Open-Source-Kunde bereits Erfahrung mit dem Produkt, das Risiko eines Fehlkaufs ist daher beträchtlich geringer – ein Vorteil für beide Seiten.

Inside Champions

In vielen Fällen existiert in der Firma des (zukünftigen) Kunden bereits ein vom Produkt überzeugter Anwender mit fachlichem Know-how (ein so genannter Inside Champion), der dem Vertrieb des Herstellers sowohl als Ansprechpartner als auch als interner Motivator seiner Kollegen agiert. Oft gelangt so freie Software sogar unter dem Radar der CIOs hindurch in Unternehmen, auch wenn dort Open-Source-Lizenzen eigentlich nicht gestattet sind [13].

Auch hier spielt der Support durch die Community eine wichtige Rolle, weil die kommerzielle Variante noch nicht vorgesehen ist und der Hersteller diesen auch nicht immer in vollem Umfang leisten könnte. Hinzu kommt, dass nur eine kleine Anzahl (Quellen sprechen von unter 1 Prozent) der Anwender später zu Kunden werden. Doch weil der User nichts für die Software bezahlt hat, ist er in der Regel gerne bereit, sich mit dem kostenlosen Community-Support zu begnügen.

Marketing und Produktmanagement

Die meisten Single-Vendor-OSS-Firmen halten es mit dem traditionellen Marketing, sie machen Werbung, treten auf Messen auf und lassen Vorträge halten [11]. Neu ist jedoch, dass eine engagierte User Community diese Aktionen unterstützt und so Geld einsparen hilft. Die freien, nicht durch Supportverträge gebundenen User liefern glaubwürdigere Empfehlungen, sie sind zufrieden mit ihren Lösungen. So wirken sie häufig als Multiplikatoren und bringen das Produkt über die Community – ohne Zutun des Herstellers – an neue Kunden.

Augustin beziffert das Verhältnis von Vertriebs- zu Entwicklungskosten in klassischen Software-Unternehmen auf 2,3 zu 1 und höher ([4], [14]). Nicht selten addieren sich die Vertriebskosten sogar auf das Sechsfache der Ausgaben für die Entwicklung (zum Beispiel bei Salesforce). Ganz anders bei Open-Source-Firmen: Die geben durchschnittlich deutlich mehr für R&D (Research and Development) als für den Vertrieb (S&M, Sales and Marketing) aus (Abbildung 3). Nicht nur aus der Sicht eines Start-ups erhöht diese Techniklastigkeit die Überlebenschancen auf dem Markt deutlich.

Abbildung 3: Während bei Open-Source-Unternehmen die Kosten für Entwicklung und Forschung die Marketing- und Vertriebsausgaben normalerweise deutlich übersteigen, ist das Verhältnis bei klassischen Softwareschmieden umgekehrt. Auffällig dabei ist die hohe Zahl an Firmen wie Salesforce, die fürs Marketing bis zu sechsmal mehr ausgeben als für Entwicklung.

Zugleich profitiert das Produktmanagement von der Innovationskraft einer Open-Source-Community. Auch wenn

der Hersteller Neuerungen erst dann in den Hauptzweig integriert, wenn der Urheber seine Rechte zu übertragen bereit ist, bringt das im Idealfall einen konstanten Fluss von Verbesserungen, sei es durch Code oder bloße Ideen.

Kein Horrorszenario: Offene Diskussionen

Die Community wird Änderungen, Verbesserungen, neue Features, Stärken und Schwächen offen diskutieren. In den Ohren klassischer Unternehmer mag das zunächst hart klingen, erweist sich auf Dauer jedoch als eine Fundgrube für die Produktmanager, die jetzt eine neue Nähe zu Anwendern erfahren und deren Ideen direkt in die Definition von Roadmaps oder kommende Featurelisten einbinden. Der Produktmanager hat hier erstmals die Chance, die Motivation und Wünsche der (Noch-)Nicht-Kunden zu erfahren und darauf einzugehen.

Die wohl größte Herausforderung bei Dual-Lizenz- oder Freemium-Modellen ist es dabei, die Kunden zur bezahlten Softwarevariante zu locken, ohne dabei die Vielzahl der freien Anwender zu vergrätzen. Erfolgreiche Projektmanager erkennen Features, die für die Open-Source-Community unwichtig sind, aber bei den Usern, die bereit sind zu bezahlen, einen großen Stellenwert haben. Die freien Anwender erhalten die Erweiterungen dann vielleicht später, mit ein paar Monaten oder Jahren Zeitversatz.

Positive Effekte für Entwicklung und Recruiting

Die Beiträge der Community beschleunigen unter Umständen aber auch die Entwicklungsarbeit. Darüber hinaus bieten sie für künftige Einstellungen einen wertvollen Ressourcenpool, mit dem der potenzielle Arbeitgeber deutlich mehr über seinen eventuellen neuen Mitarbeiter erfährt und so das Risiko einer Fehlbesetzung zu reduzieren vermag.

Doch bei Weitem am wertvollsten ist das direkte und sofortige Feedback, zum Beispiel beim gemeinschaftlichen Bugfixing in einem Daily Build. Einer schnelllebigen Developer-Community-Version (Testing) lässt sich so die stabile kommerzielle Variante gegenüberstellen, die dem professionellen Admin komfortable Sicherheit bietet. Allerdings muss die Entwicklungsleitung scharf darauf achten, dass die beiden Versionen nicht zu sehr auseinanderdriften. Redundante oder gar konkurrierende Entwicklungen sind hier zu vermeiden.

Idealfall: Self-Supporting

Eine engagierte Community leistet sich gegenseitig weitreichenden Support. Wer nicht für die Software bezahlt hat, erwartet in der Regel auch keine Unterstützung von einer Firma und ist meist bereit Community-Support in Anspruch zu nehmen, nicht selten auch solchen zu leisten. Das kommerziell aktive Unternehmen sollte hier Hilfestellung bieten, muss aber nicht den Großteil der Arbeit schultern. Der Versuch, allen Anwendern gerecht zu werden, wäre vermessen und übermäßig teuer.

Die Self-supporting Community ist vielmehr eine absolute Notwendigkeit für den Aufbau einer stattlichen Nutzerbasis, aus der sich schließlich zahlende Kunden rekrutieren lassen. Die erhalten dann Enterprise Support in Form von SLAs, Wartungsverträgen und garantierten Reaktionszeiten.

Nicht selten nutzt die Community hierbei auch dem Unternehmen, indem sie kostenlose, umfangreiche Dokumentationen, Howtos und Lösungswege für typische Fehler bereitstellt. An auch von Usern gepflegten Wikis oder Wissensdatenbanken (Knowledge Bases) führt hier kein Weg vorbei.

Fazit

Die Open-Source-Bewegung hat die Welt der Software-Entwicklung nachhaltig verändert, auch hinsichtlich der Möglichkeiten, wie Unternehmen ihre Programme zu Geld machen. Die eingangs zitierten Analysten von Gartner sind zuversichtlich, viele Beispiele von erfolgreichen Software-Projekten und -Firmen bestätigen die positive Tendenz.

Das kann jedoch nur klappen, wenn ein Unternehmen den Stellenwert der Community richtig einschätzt: Die aufzubauen und zu pflegen ist der zentrale Faktor für den Erfolg.

Infos

1. Gartner Inc., "Predicts 2009: The Evolving Open Source Model": Gartner Inc. 2008
2. Brian Fitzgerald, "The Transformation of Open Source Software": MIS Quarterly 30/03 (2006)
3. Open Source Business Models: http://robertogaloppini.net/documents/businessmodels.pdf
4. Dirk Riehle, "The Economic Motivation of Open Source: Stakeholder Perspectives": IEEE Computer 40/04 (April 2007), S. 25
5. Michael Olson, "Dual Licensing", Kapitel 5: O'Reilly 2005

6. Larry Augustin, "A New Bread of P&L: The Open Source Business Financial Model": Vortrag auf der Open Source Business Conference 2007
7. Rob Bearden, "Tailoring an Open Source Business Model": http://www.infoworld.com/event/osbc/08/
8. Don Dodge, "Freemium – Free to Paid Conversion Rates": http://dondodge.typepad.com/the_next_big_thing/2007/05/freemium_free_t.html
9. Andrew Lampitt, "Open-Core Licensing (OCL): Is this Version of the Dual License Open Source Business Model the New Standard?": http://alampitt.typepad.com/lampitt_or_leave_it/2008/08/open-core-licen.html
10. John Walker, "Building Vibrant and Sustainable Communities": Vortrag auf dem Open Source SIG of SD Forum, März 2008
11. Fabrizio Capobianco, "Building Vibrant and Sustainable Communities": Vortrag auf dem Open Source SIG of SD Forum, März 2008
12. Larry Augustin "Smoothing the On-ramp to Commercial": http://www.infoworld.com/event/osbc/08/
13. Richard T. Watson et al., "The Business of Open Source": Communications of the ACM 51/04 (April 2008), S. 41
14. Larry Augustin, "The Next Wave of Open Source: Applications": Vortrag auf der GOSCON 2005

Der Autor

Prof. Dr. Dirk Riehle ist Professor für Open-Source-Software an der Friedrich-Alexander-Universität Nürnberg-Erlangen. Er bloggt auf http://dirkriehle.com und twittert via »@dirkriehle«.