Zeitschrift: bulletin.ch / Electrosuisse
Herausgeber: Electrosuisse
Band: 96 (2005)
Heft: 7

Artikel: Weshalb Open-Source-Software entwickeln?
Autor: Zbinden, Andy
DOI: https://doi.org/10.5169/seals-857784

Weshalb Open-Source-Software entwickeln?

Zwei Open-Source-Entwickler im Gespräch

Martin F. Krafft, freiberuflicher Debian-Entwickler, und Valéry Tschopp, Informatikingenieur und SourceForge-Entwickler bei Switch, sprechen mit Andy Zbinden, Teamleiter der Switch-internen Gruppe System Engineering und Support, darüber, weshalb sie für die Open Source Community entwickeln.

Andy Zbinden

Zbinden: *Ihr entwickelt beide Software für die Open Source Community, weshalb?*

Tschopp: Switch[1)] wollte das so. Das Einfachste war, eine Open-Source-Infrastruktur wie SourceForge zu benutzen, damit alle weiterentwickeln können und damit das Projekt dann für alle zur Verfügung steht. Mehrere Entwickler in Bern und Zürich arbeiten schon daran. Es ist also einfacher, ein zentrales Concurrent Versions System Repository zu haben, das ist perfekt bei SourceForge. Man sieht, was geändert wurde.

Krafft: Ich mache das, weil ich selber auch Benutzer der Open Source Community bin. Neue Funktionalität und neue Eigenschaften sind in Open-Source-Produkten schneller implementiert als in kommerziellen Programmen. Die Grundidee ist: Ich finde eine Software gut und steuere etwas bei. Ich kann sie selber meinen Bedürfnissen anpassen, Fehler beheben, von mir gewünschte Eigenschaften einbauen und anderen zur Verfügung stellen. Ich bin nicht vom Hersteller abhängig, ich bin nicht nur auf die Funktionalität gestellt, die der Hersteller für nötig hält, sondern kann umsetzen, was ich wirklich brauche.

Tschopp: Das ist ein wichtiger Punkt. Du kannst Erweiterungen machen! Wenn dir ein Stück nicht gefällt, dann baust du dir selber ein passendes. Du musst nicht ein Mail an einen Unbekannten senden und hoffen, dass es irgendwann berücksichtigt wird. Man macht die Änderung und hat sie sofort zur Verfügung. Man kann mit den eigenen Erweiterungen weiterarbeiten und ist trotzdem kompatibel. Deshalb ist der Zugang auf das Central Repository natürlich sehr wichtig.

Zbinden: *Wie ist das denn mit der Software, wenn so viele dran arbeiten? Verästelt sie sich nicht auf unzählige Varianten?*

Krafft: Die verschiedenen Äste werden parallel entwickelt und fliessen dann wieder in die Hauptlinie ein. Die Erfahrung zeigt, dass es unwahrscheinlich ist, dass zwei Personen gleichzeitig an der gleichen Zeile feilen. Es gibt Software, die solche unabhängigen Änderungen zusammenführen und sogar Konflikte beheben kann (z.B. CVS).

Tschopp: Stimmt. Wenn man ein Feature zufügt oder ein Problem löst, ergibt das nicht eine neue Software. Die Änderung wird im neuen Release integriert sein.

Krafft: Der wichtigere Aspekt von Open Source ist die weltweite Zusammenarbeit. Dadurch wird die Software flexibel, universell, und Fehler werden rund um die Uhr gefunden.

Zbinden: *Wie sieht denn diese Zusammenarbeit aus?*

Krafft: Firmen sind lokal, man sieht sich täglich, teilt die gleichen Räume. Open Source erfordert weltweite Kooperation und Kommunikation, in einem Projekt arbeite ich zum Beispiel mit jemandem in Australien zusammen und wir kommunizieren äusserst effizient. In der Open Source Community sind Kommunikationsmedien wie Mailinglisten und IRC-Chat unverzichtbar.

Tschopp: In unserem Projekt haben wir keine Probleme mit der Kommunikation. Der Überblick bei drei oder vier Leuten ist einfach. Bei Open Source hat man generell keine Koordinationsstelle. Die Leute machen, wovon sie fühlen, dass es richtig oder wichtig ist. Was man braucht, ist Vertrauen, dass das, was die anderen hinzufügen, sich lohnt. Ein einziger Mensch könnte alles zerstören.

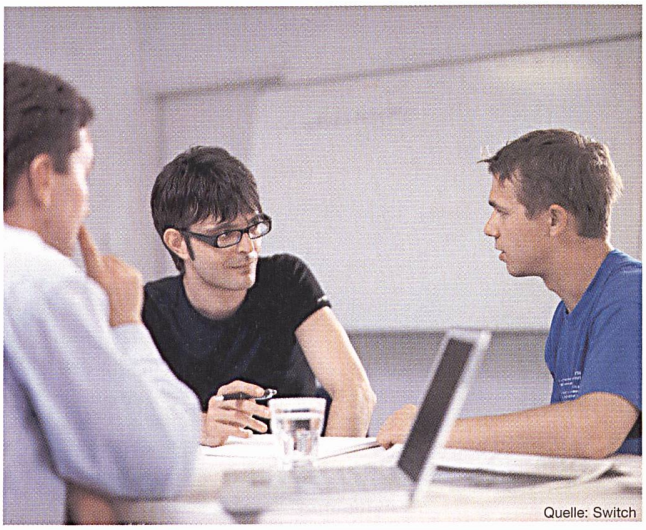
Krafft: Vertrauen ist eine grundlegende Voraussetzung, aber nicht das einzige Mittel, sonst wären wir schnell am Ende. Die Verwaltungssoftware ermöglicht uns jederzeit ein Rückgängigmachen von Änderungen.

Tschopp: Das braucht aber viel Zeit, besonders im Fall eines Ausfalls, Fehlers, von Störung, Sabotage.

Krafft: Aus diesem Grund muss man sich den Zugriff auf das Softwarearchiv erst erarbeiten, zum Beispiel indem man indirekt aushilft. Dadurch kann viel Zerstörerischem vorgebeugt werden. Wenn jemandem ein zerstörerischer Akt nachgewiesen werden kann, wird er augenblicklich rausgeschmissen. Die Versions-

Valéry Tschopp entwickelt bei Switch das AAI-Portal, das auf Shibboleth-Software basiert. AAI ist ein verteiltes Authentisierungs- und Autorisierungs-System. Er arbeitet mit zwei andern Entwicklern zusammen, das Projekt wird von SourceForge gehostet. Bei Shibboleth hat Tschopp nicht selber Zugang zum Central Repository, sondern schickt seine Erweiterungen in Form von Patches an einen Entwickler, der sie dann ins Gesamtprojekt einfliessen lässt.
tschopp@switch.ch

Martin F. Krafft ist Debian-Entwickler und -Evangelist und schrieb ein Buch mit dem Titel «The Debian System». Es richtet sich an erfahrene Administratoren, die Debian einsetzen respektive eine Entscheidungshilfe brauchen. Es erschien im Februar bei OpenSourcePress.de.
madduck@debian.org



Quelle: Switch

«Wenn dir ein Stück nicht gefällt, dann baust du dir selber ein passendes», Valéry Tschopp (Mitte) zu Open Source.

Software kann seine Änderungen eindeutig identifizieren und wieder entfernen.

Zbinden: *Wie geht das?*

Krafft: Man lässt sich die Änderungen von X in der fraglichen Zeitperiode anzeigen, geht dann jede Änderung durch und analysiert sie. Das muss ein Mensch entscheiden, die Software kann das nicht selber.

Wenn so was passiert, werden die Benutzer über die Ausmasse des «Einbruchs» informiert. Der Debian Social Contract, sozusagen unser Leitdokument, schreibt vor, dass wir keine Probleme verstecken. Wir setzen alles daran, um die Verschlechterung des Problems auszuschliessen, und informieren alle Benutzer, denn ein Benutzer möchte entscheiden können, ob er mit dem Fehler leben will oder nicht. Natürlich machen wir uns daran, den Fehler auszumerzen, meist in Zusammenarbeit mit dem Autor des betroffenen Produktes. Wir wähnen unsere Benutzer also nicht in falscher Sicherheit, wie das in anderen Firmen der Fall ist, wo man erst Monate später erfährt, dass die Software die ganze Zeit anfällig war.

Tschopp: Das ist so. Der Entwickler ist durch die Zugangsregelung zum Central Repository bekannt. Das heisst, erstens: er muss ein Stück Software zeigen, zweitens: die Leute finden es gut, drittens: er muss sich regelkonform verhalten. Die Regeln sind ungeschrieben: Ethikverständnis beispielsweise.

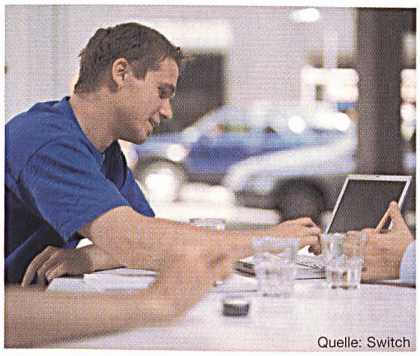
Krafft: Um Debian-Entwickler zu werden, bedarf es mehrerer Schritte. Erstens: Der Entwickler muss sich einen kryptografischen Schlüssel besorgen, den er von einem bereits eingetragenen Entwickler signieren lassen muss, unter Vorzeigen einer Identitätskarte oder eines anderen amtlichen Papiers, das seinen Namen beglaubigt. Zweitens: Er muss seine Fähigkeit unter Beweis stellen, d.h. er muss ein Debian-Paket packen, mit einem Betreuer zusammen. Es geht hier darum, dass er die Methoden und Praktiken kennt, um dann regelkonform zu sein.

Tschopp: Nicht einfach.

Krafft: Richtig. Es gibt aber Leute, die legen ein Paket vor, da muss ich sagen: «He, warum hast du dich nicht schon vor einem Jahr beworben?» Drittens: Er muss sich mit drei Dokumenten vertraut machen: dem Social Contract, den Free-Software-Richtlinien und dem Policy Manual. Diese Dokumente werden abgefragt, und wir setzen auch voraus, dass der Bewerber sich gut in der Debian-Gemeinschaft bewegen kann. All das trägt hauptsächlich zur Stärke des Betriebssystems bei.

Tschopp: Gibt es in einem der Dokumente Richtlinien über Coding Style?

Krafft: Nein. Die Fragen um den Quelltext sind jedem Projekt selbst überlassen. Es geht um die Richtlinien für das System, die Regeln, die es 10000 Paketen ermöglichen, friedlich parallel installiert zu sein. Der Prozess, einen Entwickler aufzunehmen, dauert sicherlich zwei Monate und ist nicht einfach; wir wollen schliesslich auch nur dedizierte Leute in unseren Kreis aufnehmen. Demnach ist ein Debian-Entwickler auch stolz darauf und setzt diesen Status nicht leichtfertig aufs Spiel (z.B. durch mutwillige Zerstörung). Wenn er rausgeschmissen wird, braucht er eine Vierfünftelmehrheit, die ihn wieder zulassen will. Bislang ist noch nie jemand rausgeworfen worden.



Quelle: Switch

«Ich entwickle für die Open Source Community, weil ich selber auch Benutzer bin», Martin F. Krafft.

Zbinden: *Ist das in deiner Welt anders?*

Tschopp: Sehr. Wir drei in unserem Projektteam machen kein Package, das in eine offizielle Distribution eingefügt wird. Wenn ich das Paket in die Debian-Distribution bringen wollte, müsste ich die von dir geschilderte Prozedur durchwandern.

Krafft: Oder jemanden finden wie mich, der es für dich verpackt.

Oder du verpackst es und ich «sponsere» das Paket: ich schaue es an und füge es hinzu, wenn es den Qualitätsansprüchen genügt. Aber du bist der Paketverwalter, also auch weiterhin für das Paket zuständig.

Tschopp: In jedem Fall braucht es eine Vertrauensperson als Eingangswächter.

Zbinden: *Interessantes Konzept.*

Tschopp: Die Open Source Community regelt sich selbst. Es sind sehr starke Regeln, stärker als in einer Firma.

Krafft: Zudem sind der Innovationsgrad und das Potenzial stärker als in einer Firma.

Tschopp: In einer Firmensoftware kannst du noch schnell einen Bug einfügen, wenn du gefeuert wirst. Man wird kaum Möglichkeiten haben, das rauszufinden. Wenn du in einer Firma arbeitest, traut man dir.

Krafft: Von einer Firma bist du existenziell abhängig. Du beziehst den Lohn von ihr. Bei Open Source geht es nicht um existenzielle Fragen. Da geht es um deine Ehre. Die Motivation ist da ganz anders. Viele sehen das als zu idealistisch. Aber es scheint zu funktionieren, und das mit grossem Erfolg.

Zbinden: *Ist nicht auch die physische Nähe oder Ferne des Kollegen ein Faktor? In der Firma sieht man sich ja täglich, da ist die Hemmschwelle doch grösser, etwas Schädliches zu tun, als gegenüber einem unbekannten Gesicht in Australien.*


Andy Zbinden (Mitte) leitet das Gespräch mit den Open-Source-Entwicklern Martin F. Krafft (links) und Valéry Tschopp (rechts).

Quelle: Switch

Krafft: Es sind nicht nur die unbekannten Gesichter in Australien. Es geht auch um die 10000 Entwickler, die unzähligen Benutzer, es geht um die Gemeinschaft. Auf der anderen Seite ist es ja auch mein Projekt. Ich setze es ein, es geht um mich selbst. Wenn ich dem Projekt schade, dann schneide ich mir ins eigene Fleisch. In einer Firma trägt die Firma die Last von Schäden, nicht der Einzelne. Ich kann mich nicht erinnern, dass bei Debian mal ein Fall eingetroffen ist, wo…

Tschopp: Da hat doch im letzten Jahr jemand einen Bufferoverflow in einen Befehl im Kernelmodul eingebaut. Mit gestohlenem PGP-Key. Aber 10 Leute haben gestutzt, warum das File nun geändert hat, und haben die Änderungen untersucht.

Krafft: Ist mir von Debian nicht bekannt.

Tschopp: Stimmt, da war das Linux-Kernel-Source-Projekt direkt betroffen.

Krafft: In einer Instant-Messager-Software gab es einen von früher geerbten Hackercode. Man hat das Paket dann als Ganzes entfernt. Der Entwickler der Software hat sich ins eigene Fleisch geschnitten, denn er hat die meisten Benutzer verloren, die er durch Debian hatte.

Tschopp: Das ist ein wichtiger Punkt. Die Leute sollen meine Software benutzen, denn dies ist die einzige Anerkennung, die ich bekommen kann.

Krafft: Auch. Für mich zählt aber vor allem die Resonanz, die ich erhalte. Die Benutzer sorgen dafür, dass neue Eigenschaften dazukommen und Fehler gefunden werden. So gesehen ist Open Source immer Beta-Software. Sie ist nie wirklich fertig. Man muss ja auch nichts verkaufen. Je mehr Benutzer ein Projekt hat, desto schneller kann die Entwicklung voranschreiten.

Zbinden: *Wird die Software denn immer grösser?*

Tschopp: Das ist eine Frage des Ziels. Irgendwann ist ein Ziel erreicht, dann kann es um Optimierung oder Refactoring eines Teiles gehen.

Krafft: Man hält das Projekt so einfach wie möglich. KISS – keep it short and simple. Wenn das Projekt an Featuritis leidet, geht das auf Kosten von Sicherheit und Stabilität. Das Debian-Projekt hat sich in vier Jahren verfünffacht, heute sind es 15000 Pakete. Das wird auch als Problem erkannt. Die Struktur des Software Repository muss diese Kapazität aufnehmen können.

Benutzer empfinden diese Menge an Paketen zum Teil als unangenehm; einen aus 72 Editoren auszuwählen ist ein Riesenaufwand. Der Unterschied ist dann auch nicht mehr so gross. Es gibt Überschneidungen. Und bei grösserer Anzahl an Programmen ist die Wahrscheinlichkeit von Sicherheitsproblemen dementsprechend höher.

Tschopp: Gewisse Firmen haben genau einen Editor. Der kann unmöglich die Anforderungen aller Benutzer erfüllen, auch wenn er riesengross wird. Gut ist, fünf oder sechs verschiedene Anwendungsgebiete mit fünf oder sechs verschiedenen Softwares zu bedienen. Ein einziges Monster ist schlecht, 23 Monster, die sich kaum unterscheiden, sind auch schlecht. Ein kleiner, aber guter Zoo ist viel besser.

Zbinden: *Wie sieht's denn aus mit der Kommunikationsmenge?*

Krafft: Ganz wichtig sind die Mailinglisten. Es kostet viel Zeit. Bei einer Beteiligung von 10000 Leuten könnte das eigentlich ein Vollzeitjob sein.

Tschopp: Shibboleth hat etwa 20 Mails am Tag. Auch da ist es schon schwierig, zu folgen. Das muss man fast, denn man möchte ja den Überblick behalten, um Trends zu erkennen.

Krafft: Ein Bug/Request-Tracker ist ähnlich wichtig. Die Informationen sind darin sehr strukturiert. Für die 20–30 Pakete, die ich betreue, kommen so zirka 10 Fehlermeldungen pro Woche.

Tschopp: Nicht jeder gemeldete Fehler ist ein Fehler.

Krafft: Und die Benutzer schauen nicht, ob schon jemand den gleichen Fehler gemeldet hat. Manches ist auch einfach ein Wunsch.

Tschopp: Oder der Benutzer hat etwas falsch verstanden oder falsch angewendet.

Krafft: Das ist auch ein Fehler.

Tschopp: In der Dokumentation.

Krafft: Und nicht jeder Minifehler wird zu einem neuen Release. Ich priorisiere die Fehler nach Dringlichkeit und Lust. Ein Monat Reaktionszeit für einen neuen Release ist bei meinen Projekten okay, ich habe nichts Sicherheitsrelevantes wie ssh oder libc, die sofort behandelt werden müssen.

Es ist nicht selten, dass ein anderer Entwickler oder sogar der Benutzer selbst einen Fehler in meiner Software repariert und mir die relevanten Änderungen zukommen lässt, mit der Bitte, sie einfliessen zu lassen.

Wenn es eilt, ich beispielsweise einen Fehler in ssh entdecken würde und der Verwalter von ssh ist nicht erreichbar, dann kann ich die Korrektur auch selber veröffentlichen, als so genannten NMU (Non Maintainer Upload). Der Verwalter kann dann darauf basierend seine nächste offizielle Version herausgeben.

Tschopp: Bei meinem Projekt bin ich der Hauptbenutzer. Eine To-do-Liste zum Priorisieren haben wir trotzdem, die ist sehr praktisch. Auch zum Auseinander-


«Die Open Source Community regelt sich selbst. Es sind sehr starke Regeln, stärker als in einer Firma», Valéry Tschopp (links) über die Qualität von Open-Source-Software.

Quelle: Switch

Glossar

Central Repository
Zentraler Softwarespeicher, siehe auch Concurrent Versions System

Concurrent Versions System
Ein System zur Verwaltung von Software und Softwareteilen

Debian Social Contract
Die gesellschaftliche Grundlage der Gemeinschaft für freie Software – http://www.debian.org/social_contract.html

Open Source
Freie Software, frei zugänglicher Quellcode, unentgeltlich zu benutzen, bzw. das diesem zugrunde liegende Konzept

Open Source Community
Die Gemeinschaft der Entwickler von freier Software

SourceForge
Der nach eigenen Angaben grösste Speicher für Open-Source-Software – http://www.sourceforge.net

halten von Bugs, Features, Erweiterungen und Vereinfachungen. Bei Fragen nach Erweiterungen kommt oft auch die Frage, wie aufwändig es denn sei, wann es eingebaut wird und ob überhaupt.

Zbinden: *Wo ist diese To-do-Liste? Lokal auf deiner Harddisk?*

Tschopp: Beides, lokal und ein Teil davon ist publiziert. Auf der publizierten sehen andere Leute, was ansteht, und können spontan mitarbeiten.

Zbinden: *Das mit dem Mitarbeiten ist bestechend, aber wie ist es, wenn man im Fremdteil einen Bug entdeckt? Wie schwierig ist der Code zu lesen, wenn man keine Coding-Richtlinien hat?*

Krafft: Innerhalb des Projektes gibt es schon Normen. Es ist nicht immer klug, ein C++-Projekt zum Beispiel mit Python-Extensions zu ergänzen. Dann würde der Entwickler diesen Teil an sich reissen, wenn die anderen das nicht beherrschen. Das ist unsinnig. Projektspezifische Coding-Guidelines sind meist ziemlich pedantisch, aber auch offen für Änderungen, um das Projekt dynamisch zu halten.

Tschopp: Das ist auch wichtig. Der Code muss schnell und einfach lesbar sein, mit Kommentaren versehen, mit sinnvollen Variablen-Namen. Das sind ungeschriebene Gesetze, eine Art Metagesetze.

Krafft: Aufgrund der Selbstprofilierung will der Entwickler zeigen, was er drauf hat, fast ein bisschen angeben. Wenn ich was für mich mache, dauert das nur halb so lange wie ein Open-Source-Projekt, denn Letzteres ist irgendwie wie ein Brief, den man nach der ersten Fassung in Reinschrift veredelt. Damit kann man sich einen Ruf aufbauen: «Ja, der schreibt tollen Code, mit dem arbeite ich gerne zusammen.»

Tschopp: Man zeigt seine Fähigkeiten als Entwickler. Die Qualität des Produktes ist nicht nur an der Funktionalität messbar, sondern auch daran, wie es geschrieben ist. Alle Konzepte von Softwareentwicklung sind drin, aus diesem Grunde ist eine Open Source Community möglich. Und dank dem Central Repository, und Versioning, und IRC, und Mail. Früher hatte man diese Hilfsmittel nicht, da war alles viel schwieriger.

Zbinden: *Habt ihr Wünsche an die Benutzer eurer Projekte?*

Krafft: Feedback. Konstruktives Feedback. Anhand von Feedback sehen wir, ob die Benutzer zufrieden sind. Und manchmal haben sie auch Visionen für weitere Verwendungen des Projektes. So bleibt das Produkt interessant und konkurrenzfähig.

Tschopp: Feedback. Das ist das Wichtigste. Um mehrere Gesichtspunkte zu haben, muss ich Feedback von den Benutzern bekommen: um herauszufinden, ob ich auf dem richtigen Weg bin, ob ich in die richtige Richtung gehe, ob das auch jemand braucht.

Feedback macht wirklich Spass, auch Bug-Meldungen. Dann sieht man, dass jemand die Software benutzt.

Krafft: Ich darf jedem Benutzer auch empfehlen, selbst auf Fragen in den Mailinglisten zu antworten, wenn man die Antwort kennt. Und das nicht nur, um uns Arbeit abzunehmen, sondern vor allem wegen des Lerneffektes.

Zbinden: *Auf welche Weise beziehungsweise wohin wollt ihr den Feedback?*

Krafft: debian-user@lists.debian.org ist vielleicht die beste Adresse. Generell ist es jedoch besser, erst kleinere Kreise anzusprechen und zum Beispiel debian-x anzuschreiben, wenn es sich um ein Problem mit dem X-Server handelt.

debian-user ist das Auffangbecken. Aber je spezifischer, desto besser.

Tschopp: Jedes Projekt hat an prominenter Stelle seinen Kontakt: im About oder in der Authors-Datei steht, von wem die Software entwickelt wurde und an wen man Feedback schicken kann. Jedes Projekt hat eine E-Mail-Adresse.

Zbinden: *Vielen Dank.*

Angaben zum Autor

Andy Zbinden arbeitet seit 10 Jahren bei Switch. 1998 wechselte er von der Netzwerk- in die Systemgruppe und übernahm vor gut 3 Jahren eine leitende Funktion. Heute ist er Teamleiter der Gruppe System Engineering und Support der Switch-internen IT-Dienstleistungsabteilung. Abdruck mit freundlicher Genehmigung aus dem *Switch-Journal;* Ausgabe Juni 2004.
Switch, 8021 Zürich, zbinden@switch.ch

[1] Switch betreibt das Netzwerk zwischen den Schweizer Universitäten und verbindet diese mit dem Internet. Zudem ist Switch zuständig für die Schweizer Domain-Namen.

Pourquoi développer du logiciel Open-Source?

Deux développeurs Open-Source s'entretiennent

Martin F. Krafft, développeur Debian indépendant, et Valéry Tschopp, ingénieur en informatique et développeur SourceForge chez Switch, s'entretiennent avec Andy Zbinden, du groupe interne System Engineering and Support chez Switch, sur les raisons pour lesquelles ils développent pour l'Open Source Community.