Lancierung und Verwaltung von Open Source Projekten

Open Source Projekte sind ein wichtiger Beitrag öffentlicher Forschungseinrichtungen und der Privatwirtschaft zur allgemeinen Weiterentwicklung der Informatik. Sobald die Software der Öffentlichkeit zur Verfügung gestellt wird, stellen sich Fragen der passenden Lizenzierung und der Mitwirkung von externen Entwicklern. Aus diesem Grund stellen wir in diesem Artikel die wichtigsten Open Source Lizenzen vor und zeigen in einem Fallbeispiel wie häufig benutzte Kollaborationswerkzeuge in der Open Source Software-Entwicklung gewinnbringend eingesetzt werden können.

Andreas Hofmann, Matthias Krebs, Christoph Stamm, Ursula Sury[1] | christoph.stamm@fhnw.ch

Der Begriff Open Source ist entstanden, um sich des Begriffes Freie Software zu entledigen. *Freie Software* und Open Source unterscheiden sich in der eigentlichen Bedeutung nicht wesentlich. Die Entscheidung, den Terminus Open Source zu etablieren begründete sich zum Teil auf der möglichen Missinterpretation des Wortes frei. Die *Free Software Foundation (FSF)* [FSF] verstand das Wort im Sinne von Freiheit («free speech, not free beer»), jedoch wurde es oft fälschlicherweise mit kostenlos assoziiert, da der englische Begriff *free* beide Bedeutungen haben kann.

Bruce Perens und Tim O'Reilly beschlossen, dass die Freie Software-Gemeinde mehr und besseres Marketing benötigt, um diese Art von Software weniger ideologisch zu bewerten. Der Begriff Open Source wurde von da an flächendeckend im Marketing genutzt und war auch der Namensgeber für die von Raymond, Perens und O'Reilly gegründete *Open Source Initiative (OSI)* [OSI].

Die OSI wendet den Begriff Open Source auf Software an, deren Lizenzverträge den folgenden drei charakteristischen Merkmalen[2] entsprechen:

- Der Programmcode liegt in für Menschen lesbarer und verständlicher Form vor. In der Regel handelt es sich bei dieser Form um Quelltexte in einer höheren Programmiersprache. Vor dem eigentlichen Programmablauf ist es normalerweise notwendig, diesen Text durch einen so genannten Compiler in eine binäre Form (*Binärprogramm*) zu bringen, damit das Computerprogramm vom Rechner ausgeführt werden kann. Binärprogramme sind für den Menschen im semantischen Sinne praktisch nicht lesbar.
- Die Software darf beliebig kopiert, verbreitet und genutzt werden. Für Open Source Software gibt es keine Nutzungsbeschränkungen. Weder bezüglich der Anzahl der Benutzer, noch bezüglich der Anzahl der Installationen. Mit der Vervielfältigung und der Verbreitung von Open Source Software sind auch keine Zahlungsverpflichtungen gegen eine Lizenzgeberin verbunden.
- Die Software darf verändert und in der veränderten Form weitergegeben werden. Durch den offen gelegten Quelltext ist Verändern ohne weiteren Aufwand für jedermann möglich. Weitergabe der Software soll ohne Lizenzgebühren möglich sein.

Gerade der dritte Punkt zeigt deutlich, dass Open Source Software förmlich von der aktiven Beteiligung der Anwender an der Entwicklung lebt. So bietet sich Open Source Software zum Lernen, Mitmachen und Verbessern an. In den Abschnitten *Lancierung eines Open Source Projektes* und *Fallbeispiel* zeigen wir zuerst allgemein und dann anhand eines konkreten Beispiels von einer Programmbibliothek zur Bildkomprimierung auf, welche Schritte notwendig sind, um selber ein Open Source Projekt zu lancieren und zu verwalten.

Trotz den drei genannten charakteristischen Merkmalen scheint der Begriff Open Source nicht frei von Interpretationen zu sein, da selbst die FSF und die OSI sich nicht immer einig darüber sind, ob die Nutzungsrechte einer Software-Lizenz den von FSF bzw. OSI verlangten Kriterien von Open Source genügen. Der Begriff der Software-Lizenz und die Anwendung einer solchen sind hier sehr zentral und werden daher im nachfolgenden Abschnitt genauer dargestellt.

Software-Lizenzen

Eine Lizenz ist ein Gebrauchsrecht für etwas Nichtmaterielles. Im Falle einer Software-Lizenz wird die Art und Weise des erlaubten Gebrauchs der Software klar geregelt. Software-Lizenzen treten nicht nur im Open Source Umfeld auf, son-

1 Ursula Sury ist selbständige Rechtsanwältin und Dozentin an der Hochschule Luzern.

2 Diese drei Charakteristika sind in der Open Source Definition der Open Source Initiative detailliert festgelegt.

dern immer bei der Benutzung von Software. Wer Software einsetzt, akzeptiert global – entweder stillschweigend oder ausdrücklich durch Bestätigung die Lizenzvereinbarungen gelesen zu haben – die allgemeinen Geschäftsbedingungen zwischen Lizenzgeber (z. B. Software-Hersteller) und Lizenznehmer (Benutzerin). Da die Benutzer in den meisten Fällen blind vertrauen, sind die Benutzer gegen abstruse Geschäftsbedingungen grundsätzlich geschützt.

Mehr als blindes Vertrauen ist bei der Lancierung eines eigenen Open Source Projektes angebracht. Denn für das eigene Projekt sollte eine passende Software-Lizenz ausgewählt werden und dies setzt wiederum voraus, dass verschiedene Lizenzen bekannt sind. Aus diesem Grund listen wir in Abschnitt *Verschiedene Open Source Lizenzen* ein paar wichtige Open Source Software-Lizenzen auf und stellen sie in Abschnitt *Auswahl einer passenden Lizenz* einander gegenüber. Doch zuerst erklären wir die Begriffe Lizenz, Copyright und Copyleft.

Bei all diesen Ausführungen betonen wir, dass wir die Sachlage aus der Sicht von Informatikern betrachten und uns für einfachen Umgang mit Wissen einsetzen. Des weitern ist wichtig zu erwähnen, dass beim weltweiten Austausch von Software verschiedene Rechtssysteme aufeinander prallen. Lizenzbedingungen werden meistens vor dem Hintergrund des Rechtssystems desjenigen Staates formuliert, in dem die Software entwickelt wird. Die Anwendung dieser Lizenzbedingungen in einem fremden Rechtsystems ist dann eine ganz andere Sache. Bei unseren Betrachtungen gehen wir vom Schweizer Recht [CHLR] aus, wo nichts Gegenteiliges erwähnt ist, welches im Bereich des Urheberrechtes weltweit ähnlich ist.

Lizenz

Treu der Open Source Philosophie zitieren wir Wikipedia [Wiki] zum Begriff Lizenz, da der ganze Inhalt von Wikipedia unter der *GNU*-Lizenz für freie Dokumentation [GNU] steht, die das Kopieren von Texten für jegwelche Zwecke ausdrücklich erlaubt:

> *Allgemein ist eine Lizenz (v. lat.: licere = erlauben) eine Erlaubnis, Dinge zu tun, die ohne diese verboten sind. Dies können einerseits staatlich erteilte Sonderrechte sein, zum Beispiel in der ehemaligen DDR die Spielerlaubnis für Musiker oder die «Lizenz zum Töten» des fiktiven Agenten James Bond, andererseits, bei gewerblichen Schutzrechten wie dem Urheber- oder Patentrecht, von juristischen oder natürlichen Personen mittels Verträgen eingeräumte Rechte.*

Es geht also um eine Erlaubnis, um ein Recht, im Fall von Software-Lizenzen um ein Recht auf Verwendung von Software (ähnlich der Miete eines Produkts). Werden Computerprogramme in einem Arbeitsverhältnis in Erfüllung vertraglicher Pflichten geschaffen, so ist die Arbeitgeberin allein zur Ausübung der ausschliesslichen Verwendungsbefugnisse berechtigt. Ansonsten besitzen die Urheber das ausschliessliche Recht auf Gebrauch. Generell wird hier zwischen einfachen und ausschliesslichen Rechten unterschieden. Wenn eine Urheberin dem ausschliesslichen Gebrauch ihres Werkes durch jemand anderen zustimmt und sich keinerlei Gebrauchsrechte vorbehält, so kann der neue Rechtsbesitzer unter Ausschluss aller, einschliesslich der Urheberin selber, das Werk exklusiv nutzen. An der Urheberschaft ändert sich aber nichts. Man spricht hier von einem ausschliesslichen Recht. Der Besitzer eines solchen Rechtes kann einfache Gebrauchsrechte an Dritte vergeben, wobei die Urheberin damit einverstanden sein muss. Der Besitzer eines einfachen Gebrauchsrechtes ist nur dazu berechtigt, das Werk auf die vertraglich festgelegte Weise zu gebrauchen.

Bei freier Software werden einfache Gebrauchsrechte pauschal an jedermann eingeräumt. Die Zustimmung des Lizenznehmers wird üblicherweise stillschweigend durch sein korrektes Verhalten in der Anwendung der Lizenz signalisiert; es muss nichts unterschrieben werden. Insofern der Rechteinhaber keine Gegenleistungen fordert, kann die Lizenz kurz und simpel ausfallen. Ein bekanntes Beispiel lautet: «Do the fuck you want with it.» – ein wenig formlos, doch juristisch gültig. Problematisch ist jedoch bei solchen Lizenzen, dass veränderte Versionen der Computerprogramme nach geltendem Recht nicht automatisch ebenso freigiebig an jedermann lizenziert sind. So geniessen so genannte Werke zweiter Hand ihren eigenständigen Schutz, so lange nichts anderes vereinbart worden ist. Die *GNU*[3] *General Public License* [GPL] zum Beispiel versucht daher die Freiheiten zu bewahren und verlangt eine Gegenleistung für die eingeräumten Rechte:

- Dass das Programm nicht allein als Binärprogramm, sondern nur zusammen mit einer für Menschen verständlichen Version, dem Quelltext, weitergegeben wird.
- Dass veränderte Versionen nur dann verbreitet werden dürfen, wenn sie ebenfalls unter die GPL gestellt werden. Wer sich nicht daran hält, verliert seine eigenen Gebrauchsrechte wieder.

Diese Verfahrensweise, eine Freiheit zu bewahren, wird in Anlehnung an den Begriff Copyright Copyleft genannt.

Copyright

Das Copyright stammt aus dem angloamerikanischen Raum und ist dem Schweizerischen Urheberrecht [URG] ähnlich. Es unterscheidet

3 GNU ist ein rekursives Akronym von GNU's not Unix.

sich aber in der Ausrichtung insofern, dass das Copyright eher die wirtschaftlichen Aspekte (den Schutz von wirtschaftlichen Investitionen), hingegen das Urheberrecht in der Schweiz mehr die Urheberin eines Werkes als Schöpferin und deren ideellen Beziehungen zum Werk berücksichtigt. Das angloamerikanische Copyright wird nicht automatisch der Urheberin eines Werkes zugestanden, sondern es kann einem wirtschaftlichen Rechteverwerter (z. B. Verlag) übertragen werden. Im Gegensatz dazu wird das Schweizerische Urheberrecht der Schöpferin ohne weiteres Zutun automatisch zugesprochen. Die Urheberin ist jedoch frei ihr Urheberrecht an eine andere natürliche Person zu übertragen. Damit ein Werk urheberrechtlich geschützt sein kann, ist eine geistige Schöpfung mit individuellem Charakter notwendig. Zudem muss es eine gewisse Schöpfungshöhe aufweisen; triviale Werke fallen somit nicht unter das Urheberrecht. Computerprogramme sind im URG explizit als Werke aufgelistet.

Copyleft

Für Fortentwicklungen gilt wie für Erstentwicklungen, dass unter Einbezug der Schöpfungshöhe ein eigenständiges Urheberrecht besteht. Ist das ursprüngliche Werk zum Beispiel für jeden frei kopierbar, verteilbar, veränderbar usw., so übertragen sich diese Freiheiten jedoch nicht automatisch auf die Fortentwicklungen des Werks. Dadurch kann freie Software, welche nicht speziell lizenziert wird, zum Ausgangsmaterial künstlich knapper *proprietärer Software*[4] werden. Da dies oft nicht erwünscht ist, beinhalten gewisse Lizenzen eine Copyleft-Klausel.

Gemäss Wikipedia ist Copyleft ein Schutzverfahren in bestimmten Lizenzen freier Software beziehungsweise freier Inhalte (z. B. Software), welches einen bestimmten Aspekt des Copyrights (beziehungsweise Urheberrechts) in sein Gegenteil zu verkehren versucht; daher auch der Name. Das heisst, mit einer Copyleft-Vereinbarung wird die Freiheit von Weiterbearbeitungen und Fortentwicklungen eines freien Werkes erzwungen, um dadurch die Vereinnahmung zu verhindern. So erlaubt die Urheberin einer Software beispielsweise die freie Verwendung und die Weiterentwicklung ihres Quelltextes nur dann, wenn der Urheber der Fortentwicklung sich damit einverstanden erklärt, seine Weiterentwicklung unter den gleichen Bedingungen wie das ursprüngliche Werk frei zu geben.

Nicht alle Entwickler freier Software sind jedoch mit den gängigen Copyleft-Vereinbarungen einverstanden. Durch den Zwang, Fortentwicklungen ebenfalls unter die gleiche Lizenz zu stellen, sehen sie die Freiheit unangemessen eingeschränkt und raten stattdessen zu Copyleft-freien Lizenzen.

Eine Copyleft-Klausel ist oft auch sinnvoll, weil sich die Urheberrechtsbestimmungen zwischen Werken zweiter Hand und Werken mit mehreren Miturhebern dahingehend unterscheiden, dass im Fall eines Werks zweiter Hand, die Urheberin des bestehenden Werks kein Mitsprache bei der Nutzung des Werks zweiter Hand erhält, dass aber bei Werken mit Miturhebern allen gemeinsam die Urheberschaft und somit ein Mitspracherecht zusteht.

Problematisch beim Copyleft ist, dass zwei verschiedene Copyleft-Lizenzen oft miteinander inkompatibel sind. Das heisst, es können keine zwei Werke mit verschiedenen Copyleft-Lizenzen zu einem einzigen kombiniert werden. Aufgrund der grossen Verbreitung der GPL in Zusammenhang mit Computerprogrammen ist der Fall zweier unterschiedlicher Copyleft-Lizenzen jedoch eher selten. Für freie Literatur, freie Musik usw. lauert hier aber eine grosse Gefahr. Solche Inkompatibilitäten können nur dann aufgelöst werden, wenn die Lizenzen selbst erlauben, dass das Werk unter eine andere Lizenz gestellt werden darf (z. B. GPL bei der LGPL Version 2.1) oder sich selbst als Ausnahmen einer bestehenden Lizenz definieren (z. B. LGPL Version 3 als Ausnahme der GPLv3). Die bekanntesten Lizenzmodelle, die die Open Source Bedingungen gemäss der OSI erfüllen, sind nachfolgend aufgeführt. Dabei sind die GPL und die LGPL die mit Abstand am weitesten verbreiteten Lizenzmodelle [OSL, IFROSS].

GNU General Public License (GPL)

1983 legte Richard Stallman, Computerpionier am MIT, den Grundstein des GNU-Projekts. GNU sollte das proprietäre UNIX-Betriebssystem (bestehend aus einem Kernel und verschiedenen Anwenderprogrammen) durch freie Software ersetzen. Es sollte jedem gestattet sein, den Code der Software frei zu verändern und weiterzugeben. Eine kommerzielle Nutzung war dennoch nicht ausgeschlossen; so verkaufte Richard Stallman selbst frühe Versionen der GNU-Software auf Datenträgern.

1989 entwickelte das GNU-Projekt für seine Software eine einheitliche Lizenz, zur Verankerung der folgenden Freiheiten:

- Freiheit, das Programm für jeden Zweck auszuführen;
- Freiheit, den Quelltext des Programms zu studieren und anzupassen;
- Freiheit, das Programm zu kopieren;
- Freiheit, das veränderte Programm zu kopieren.

Diese Freiheiten sollten auch langfristig sichergestellt werden. Zu diesem Zweck enthält die GPL zwei wesentliche Klauseln, welche das Programm selbst als auch vom Quelltext abgeleitete Varianten (Derivate) betrifft:

4 Unter proprietärer Software verstehen wir Software, die keine Open Source Software ist.

- Jedes Derivat muss ebenfalls vollständig unter der GPL lizenziert werden;
- Bei Weitergabe des Binärprogramms muss der Quelltext des gesamten Programms mitgeliefert oder auf Anfrage ausgehändigt werden.

Die Anwendung der GPL auf eine Programmbibliothek führt dazu, dass jedes Programm, welches diese Bibliothek nutzt (gleich ob statisch dazu gebunden oder dynamisch zur Laufzeit aufgerufen) ebenfalls unter der GPL stehen muss. Eine eigene Problematik ist die Zusammenstellung von GPL- und nicht GPL-lizenzierten Werken, auf welche wir hier nicht weiter eingehen.

Das GNU-Projekt erwies sich als recht erfolgreich. Heutige Linux-Systeme beruhen zu grossen Teilen auf GNU-Software (z. B. GNU C-Compiler, GNOME-Desktop). Schon aus dieser Tradition heraus verwenden viele neue Open Source Projekte die GPL. Die GPL garantiert, dass freie Software stets freie Software bleibt und dass Software-Firmen nur dann GPL-lizenzierte Software in eigenen Produkten einsetzen und verkaufen, wenn sie den Quelltext ihrer Software frei zugänglich machen.

GPLv3

Die dritte und neuste Version der GNU General Public License wurde am 29. Juni 2007 verabschiedet. Sie löst damit die seit 16 Jahren bestehende Version 2 ab. Mit den Anpassungen soll die GPLv3 [GPLv3] den heutigen Begebenheiten in der Softwarelizenzwelt besser Rechnung tragen.

Ein oft auftretendes Problem bei juristischen Formulierungen ist, dass gewisse Begriffe je nach Land unterschiedlich interpretiert werden, was dazu führen kann, dass die Lizenzbedingungen unterschiedlich ausgelegt werden. Diesem Umstand trägt die GPLv3 vermehrt Rechnung, indem sie alle Begriffe, welche missverständlich interpretiert werden können, genauer erklärt. So wird beispielsweise neu der Begriff «convey» anstatt dem bisherigen «distribute» für die Verbreitung einer Software verwendet [QG].

Die GPLv3 passt sich aber auch an neue Copyright-Begebenheiten an, welche in den letzten Jahren unter anderem im Rahmen des Digital Millennium Copyright Act (DMCA) Einzug gehalten haben. Der DMCA untersagt zum Beispiel die Umgehung technischer Schutzmassnahmen wie DRM (Digital Rights Management) in Werken. Anders als die GPLv2, welche DRM generell ablehnt, erlaubt die GPLv3 DRM-Funktionalität in GPL-lizenzierten Quelltext einzubauen. Gleichzeitig deklariert die GPL jedoch ein DRM, welches mit GPL-lizenziertem Quelltext implementiert wurde, als eine technisch nicht wirksame Schutzmassnahme, weshalb der DMCA in solch einem Fall ungültig und damit die Umgehung des DRM erlaubt ist.

Neben dem DMCA, welcher zu grosser Aufregung geführt hat, sind in letzter Zeit einige Probleme mit Software-Patentklagen in den USA aufgetreten. Um Benutzer und Entwickler von GPL-basierter Software vor solchen Patentklagen zu schützen, schreibt die GPLv3 vor, dass kein Lizenznehmer gegen einen Anderen eine Patentklage einreichen darf, so lange dieser den GPL-Code im Rahmen der Lizenz nutzt oder weiterentwickelt.

Des weitern bietet die GPLv3 das uneingeschränkte Recht auf die Nutzung modifizierter Software auf einem Gerät, welches mit GPL-lizenzierter Software ausgestattet ist. Eine sogenannte *Tivoization*[5] ist verboten. Grundsätzlich ist eine Verschlüsselung oder eine Integritätsprüfung zum Schutz des kompilierten Programms zulässig, allerdings muss auf Verlangen die Information, die zur Entschlüsselung nötig ist, herausgegeben werden.

Die dritte Version der GPL zeichnet sich zudem durch eine bessere Lizenzkompatibilität aus. Eine Fremdlizenz ist dann mit GPLv3 kompatibel, wenn die Fremdlizenz ausschliesslich Zusatzparagraphen enthält, welche die GPL nicht lockern, falls sie weggelassen werden. So kann zum Beispiel in einer Fremdlizenz die Verwendung geschützter Markenzeichen in einem zusätzlichen Paragraphen explizit untersagt sein. Eine solche Fremdlizenz ist dennoch zur GPLv3 kompatibel, denn die Verwendung geschützter Markenzeichen wäre auch ohne diesen Paragraphen unzulässig.

Mozilla Public License (MPL)

Als im Februar 1998 der Quelltext des Web-Browsers Mozilla freigegeben wurde, stellte das Mozilla-Projekt gleich auch eine neue Software-Lizenz [MPL] vor, die Mozilla Public License Version 1.0. Mittlerweile wird die revidierte Version 1.1 verwendet. Die MPL enthält einige neuartige Klauseln, ähnelt aber grundsätzlich der LGPL.

Unterschieden wird zwischen Dateiderivaten und Werkderivaten. Dateiderivate sind Änderungen an einzelnen MPL-lizenzierten Dateien, ihre Zusammenführung oder Inklusion in anderen Dateien. Werkderivate sind Werke, die Funktionen aus den MPL-lizenzierten Dateien aufrufen oder von ihnen aufgerufen werden. Der Quelltext von Dateiderivaten muss auf Anfrage ausgehändigt werden. Dateiderivate müssen ebenfalls unter der MPL lizenziert werden. Werkderivate können dagegen beliebig lizenziert werden.

Damit wird sichergestellt, dass die Schnittstellen der Applikation offen bleiben, einzelne Erweiterungen jedoch proprietär sein können. Diese Vorgehensweise ist sicherlich besonders

5 Benannt nach dem Hardware-Mediaplayer TiVo, welcher trotz GPL-lizenziertem Quelltext die Ausführung modifizierter Software mittels Integritätscheck verhinderte.

bei einem Web-Browser nachvollziehbar, aber auch für andere modulare Systeme geeignet.

Zwar schreibt die MPL für Dateiderivate die Aushändigung des Quelltexts vor, erlaubt aber für die Binärdateien beliebige Lizenzierung, sofern die Lizenzbedingungen nicht mit der MPL in Konflikt stehen. So kann z. B. die Verbreitung von bestimmten Binärdateien untersagt werden, die Herstellung und freie Verbreitung von Binärdateien durch Dritte jedoch nicht.

Die MPL enthält auch eine salvatoresche Klausel, was bedeutet, dass die verbleibenden Bedingungen auch rechtskräftig sind, wenn es einzelne Bedingungen aufgrund im Lande des Rechtsvorgangs geltender Bestimmungen nicht sind. So könnten z. B. Kryptographie-Exportverbote eine Verbreitung bestimmten Quelltexts verhindern, dies würde jedoch die übrigen MPL-Bedingungen nicht tangieren, sie müssen soweit möglich erfüllt werden.

Seit Version 1.1 erlaubt es die MPL, einzelne Dateien des Werkes unter mehreren Lizenzen zur Verfügung zu stellen, was bei Wahl einer GPL-kompatiblen Lizenz auch die Verbindung von GPL- mit MPL-lizenziertem Quelltext ermöglicht.

GNU Lesser General Public License (LGPL)

Möchte man den Open Source Gedanken so weit wie möglich unterstützen, sollte man eine Copyleft-Lizenz wählen, also z. B. GPL oder MPL. Möchte man dagegen nicht verhindern, dass proprietäre Programme den eigenen Quelltext nutzen und womöglich gewinnbringend vermarkten, wie dies beispielsweise für die Etablierung eines De-Facto-Standards mithilfe von Programmbibliotheken oft notwendig ist, so kann man sich für eine Copyleft-freie Lizenz entscheiden. Die grosse Verbreitung einer neuen Technologie wie z. B. des Bildformates PNG [PNG] hängt nicht unwesentlich davon ab, dass verschiedenste Software-Hersteller die Programmbibliothek libpng integrieren und dadurch die Bildformate der PNG-Familie in ihren Produkten unterstützen.

Für Programmbibliotheken bildet die LGPL einen speziellen Kompromiss, der den Copyleft-Effekt aufrechterhält, aber gleichzeitig die Nutzung innerhalb proprietärer Software nicht unnötig erschwert.

Ursprünglich hiess die LGPL «Library General Public License», sie wurde jedoch in «Lesser» umbenannt, da Richard Stallman befürchtete, sie würde sonst zu häufig für Programmbibliotheken verwendet. Der Begriff Lesser soll kenntlich machen, dass die Nutzungsfreiheit (im Sinne des GNU-Projekts) pragmatischen Motiven untergeordnet wird. Die LGPL unterscheidet sich von der GPL im Wesentlichen dadurch, dass sie sowohl die statische als auch die dynamische Bindung einer LGPL-lizenzierten Bibliothek mit beliebigen Programmen erlaubt, ohne dass automatisch das gesamte Programm unter den Bedingungen der Lizenz [LGPL] stehen müsste. Allerdings muss ein Reverse Engineering des gesamten Programms gestattet sein.

Die LGPL erzwingt gleichzeitig das «Copyleft» für den Code der eigentlichen Programmbibliothek: Alle Veränderungen an der Bibliothek selbst müssen wiederum frei verfügbar sein. Weiter wird von dem die Bibliothek verwendenden Programm verlangt, dass es auf die Benutzung der Bibliothek an prominenter Stelle deutlich aufmerksam macht.

BSD License

Die BSD-Lizenz hat eine andere historische Tradition als die GPL bzw. LGPL. Der Name stammt von der Berkeley Software Distribution, einer seit den 1970er Jahren entwickelten UNIX-Distribution (BSD-Lite), die Anfang der 1990er als Open Source Software zur Verfügung gestellt wurde. Daraus entwickelten sich verschiedene freie UNIX: OpenBSD, NetBSD, FreeBSD usw. Die Lizenzbedingungen [BSD] sind einfach:

- Das Programm darf in jeder Form, auch in Binärform, weitergegeben werden. Eine Pflicht zur Überlassung des Quelltexts besteht nicht;
- Bei der Weitergabe in Binärform muss die Lizenz den Dateien beigefügt werden;
- Bei Derivaten darf der Name der Autoren bzw. des Herstellers nicht ohne Erlaubnis für Werbezwecke verwendet werden.

Zusätzlich enthielt die erste Version der BSD-Lizenz eine so genannte Werbeklausel, die es erforderlich machte, in Werbematerialien auf die Universität Berkeley hinzuweisen. Diese Klausel wurde 1999 aufgrund von Beschwerden aus den originalen Quellen von BSD-Lite entfernt. Das FreeBSD übernahm diese Lizenzänderung grösstenteils für die eigenen Erweiterungen, NetBSD und OpenBSD folgten dem nicht. Die erste Version der BSD-Lizenz wird dennoch von vielen Open Source Projekten verwendet. So finden sich in allen BSD-Versionen (und bei anderen, die BSD-lizenzierte Software vertreiben) etliche Dutzend dieser Klauseln, mit dem Hinweis auf den jeweiligen Verfasser.

Apache Software License

Die Apache Software Foundation, die von namhaften IT-Unternehmen unterstützt wird, hat für ihre Open Source Serverprodukte (am bekanntesten ist der Apache Webserver) eine eigene Lizenz entwickelt. Diese Lizenz erlaubt Veränderungen am Quelltext und die ausschliessliche Weitergabe in Binärform, sofern die Lizenzbedingungen [Apache] beigefügt werden. Sie enthält eine BSD-ähnliche Werbeklausel, die in Version 1.1 auf Dokumentation beschränkt wurde. Sie verbietet ausserdem den Gebrauch des Namens Apache für Derivate.

Die Apache Software-Lizenz ist sehr stark auf Apache-Produkte spezialisiert und empfiehlt sich kaum für allgemeine Open Source Applikationen.

Auswahl einer passenden Lizenz

Die zuvor aufgelisteten Software-Lizenzen sind nur eine kleine, bekannte Auswahl. Selbstverständlich steht es jedem Entwicklungsteam offen, eine eigene Software-Lizenz zu verfassen. Während grosse Anbieter diesen Aufwand nicht scheuen, so greifen kleinere Entwicklungsteams üblicherweise auf bestehende Lizenzen zurück. Dabei erfordert die grosse Verschiedenheit dieser Lizenzen, dass bei der Wahl einer passenden Lizenz für ein eigenes Open Source Projekt, mehrere Aspekte beachten werden.

So sollte man sich zunächst überlegen, welche Fremdbibliotheken, -module oder welchen fremden Quelltext man in der eigenen Software verwendet und unter welchen Bedingungen diese publiziert worden sind. Wird fremder Open Source eingesetzt, so ist die Wahl der eigenen Lizenz stark eingeschränkt, da die Lizenzen untereinander kompatibel sein müssen. Verwendet man hingegen keine anderen Open Source Projekte, so ist man frei in der Wahl und muss sich lediglich über die angestrebte Nutzung der eigenen Software im Klaren sein.

Die zuvor besprochenen Lizenzen unterscheiden sich hauptsächlich im Copyleft-Effekt, d.h. in der Pflicht, Erweiterungen und Verbindungen zu anderer Software ebenfalls unter eine Open Source Lizenz zu stellen. Die BSD-Lizenz ist in dieser Hinsicht die liberalste, die GPL die restriktivste. In Tabelle 1 fassen wir diesen Sachverhalt zusammen.

Die GPL, die beinahe schon als Standardlizenz gilt, macht meist am wenigsten Probleme.

Lizenz	Freiheits-grad	Copyleft-Effekt		Urheber-Hinweis
		Quelltext verwenden	Bibliothek einbinden	
BSD	hoch	beliebige Lizenz	beliebige Lizenz	in Derivaten keine Werbung mit dem Urheber
MPL	mittel	MPL	beliebige Lizenz	in Derivaten Hinweis auf die Lizenz
LGPL	mittel	LGPL	beliebige Lizenz	in Derivaten Hinweis auf die Lizenz an prominenter Stelle
GPL	gering	GPL	GPL	Hinweis auf Lizenz in allfällig vorhandenem Lizenzdialog

Tabelle 1: Vergleich der wichtigsten Lizenzen bezüglich Freiheitsgrad, Copyleft-Effekt und notwendiger Hinweise auf die Urheberschaft.

Der wesentliche Unterschied zu vielen anderen Lizenzen ist das Verbot der Weitergabe in der Binärform ohne die Bereitstellung des Quelltextes. Als Urheber hat man stets die Möglichkeit, die Lizenz einer Software zu ändern, was im Fall von freigegebenen Inhalten jedoch nicht unproblematisch ist, da die veränderten Lizenzbedingungen nur für neue Lizenznehmer gelten. Faktisch führt dies dazu, dass Rechte an bestehendem Quelltext nicht eingeschränkt werden können. Es ist aber durchaus möglich, ab einem bestimmten Stand des Open Source Projektes, den neu entwickelten Quelltext unter neue Lizenzbedingungen zu stellen. Wird von einem Urheber diese Möglichkeit wahrgenommen, mag sich jedoch der eine oder andere aktive Open Source Mitentwickler vor den Kopf gestossen fühlen.

Anwendung von Lizenzen

Eine Lizenz anwenden ist einfach: Die Lizenz sollte als Datei dem Programm beigefügt werden – in der Unix-Welt hat sich der Dateiname COPYING eingebürgert. Jede Quelltext-Datei sollte in kurzer Form auf den oder die Urheber, den Namen der Lizenz, die Version und die Datei verweisen, also z. B.:

```
Diese Datei ist Bestandteil von
MegaCalc, 1994-2003 entwickelt von
Werner Muster, Klaus Beispiel und
anderen, siehe die Datei CREDITS für
Details. MegaCalc ist freie Software
unter den Bedingungen der GNU
General Public License Version 2.
Die Haftung ist ausgeschlossen.
Diesem Programm sollte die Datei
COPYING beiliegen, welche die
Lizenzbedingungen enthält. Sollte
dies nicht der Fall sein, können Sie
die Lizenz bei der Free
Software Foundation (www.fsf.org)
anfordern: Free Software Foundation,
Inc., 59 Temple Place, Suite 330,
Boston, MA 02111-1307 USA.
```

Zusätzlich empfiehlt es sich und bei bestimmten Lizenzen ist es sogar erforderlich, die Lizenzbedingungen beim Programmstart oder bei der Installation mindestens einmal einzublenden.

Bei mehrsprachiger Software sollte der Hinweis in englischer Sprache eingefügt werden. Anwendungsbeispiele für die einzelnen Lizenzen finden sich in einer Liste von Open Source Lizenzen der OSI [OSIa].

Lancierung eines Open Source Projektes

Was bedeutet es, ein Open Source Projekt zu lancieren und es zu unterhalten? Welche Verantwortungen und Aufgaben gegenüber der grossen Internetgemeinde nimmt man dadurch auf sich? In diesem Abschnitt wollen wir versuchen, diese Fragen zu klären und zu zeigen, dass die Initiierung eines Open Source Projektes kein Hexenwerk ist und dass sich der Unterhalt des

Projektes in Grenzen hält. Durch Verwendung von vorhandenen Open Source Plattformen kann einerseits viel Eigenarbeit gespart und anderseits der Projektfortschritt beschleunigt werden.

Loslassen

Einer der schwierigsten Punkte bei der Lancierung eines Open Source Projektes ist vermutlich der, dass der eigene Programmcode aus der Hand gegeben werden muss. Das erfordert eine gewisse Bereitschaft und Offenheit, denn nach der Veröffentlichung kann nicht mehr selbst bestimmt werden, was mit dem Quelltext alles gemacht wird. Es bestehen aber immer noch genügend Möglichkeiten, beim lancierten Projekt die wichtigen Dinge zu entscheiden. Die Kontrolle darüber, welcher Quelltext und welcher nicht zum Projekt beigesteuert wird, wird nicht abgegeben.

Hat man sich erst einmal damit abgefunden, dass der Quelltext ab jetzt allen zugänglich ist, so sollen auch die Möglichkeiten genutzt werden, die ein Open Source Projekt bietet. Primär geht es um die potentiell grosse Zahl an möglichen «Mitarbeitern». Sehr bald werden einzelne, speziell interessierte Benutzer aus der Masse hervorstechen. Diese gilt es einzubinden. Meist haben solche Benutzer oder eben auch Entwickler am Inhalt des Projektes ein eigenes, grosses Interesse und sind selbst daran interessiert, Verbesserungen oder Erweiterungen am Projekt vorzunehmen. Damit diese Entwickler das nicht im stillen Kämmerlein für sich selbst tun und das Projekt nicht davon profitiert, muss man dafür sorgen, dass engagierte Entwickler ihre Leistungen ins Projekt einbringen können. Verlässlichen und interessierten Entwicklern soll daher Zugriff auf die Quelltext-Verwaltung gegeben werden. Einerseits können diese so ihre Entwicklungen dem Projekt beisteuern, andererseits können sie einem auch Arbeit abnehmen, indem sie z.B. rapportierte Fehler beheben. Es darf von der Nutzergemeinde durchaus auch etwas erwartet werden, und die Bereitstellung des eigenen Quelltexts in einem Open Source Projekt muss nicht als rein selbstloser Akt betrachtet werden.

Vorabklärungen

Ist die Projektidee für ein Open Source Projekt vorhanden, oder hat man sich entschieden, ein bestehendes Projekt als Open Source frei zu geben, so muss eine geeignete Software-Lizenz ausgewählt und eine Plattform für die Entwicklung, Pflege und Verwaltung des Open Source Projektes bestimmt werden. Bei der Wahl der Plattform stehen grundsätzlich zwei Möglichkeiten zur Verfügung: Aufsetzen einer eigenen oder Auswahl einer bestehenden.

Der Aufwand für die Wahl einer passenden Lizenz und einer komfortablen Plattform hängt stark von den Vorkenntnissen und den gesammelten Erfahrungen mit Open Source Projekten ab. Ist überhaupt kein Vorwissen vorhanden, und muss man sich zuerst über Lizenzen, Plattformen usw. einlesen, so kann dieser Arbeitsschritt bis zu einer Arbeitswoche beanspruchen.

Eine geeignete Plattform beinhaltet Client-Server-Werkzeuge zur verteilten Software-Entwicklung und zur Koordination der Abläufe zwischen den beteiligten Personen. Dazu gehören üblicherweise auch Funktionen zur Verwaltung der Projektmitglieder.

Neben diesen einzelnen Werkzeugen existieren auch komplette Open Source Plattformen, welche die zuvor genannten Dienste allesamt anbieten. Solche Open Source Plattformen bieten darüber hinaus eine erhöhte Publizität, da das eigene Projekt vom Bekanntheitsgrad der Plattform selber profitieren kann. Als weiterer Vorteil kommt hinzu, dass man sich nicht selbst um die Hardware- und Softwareinfrastruktur kümmern muss.

Eine Auswahl an Plattformen

Zur Führung von Open Source Projekten stehen einige komplette Plattformen zur Auswahl. Nicht alle sind gleich populär und nicht alle bieten die genau gleichen Dienste an.

Die Entwicklung des GNU-Projektes läuft auf *Savannah* [Sav]. Derzeit werden dort etwa 2800 Projekte verwaltet, an denen mehr als 50'000 registrierte Benutzer teilnehmen. Neben den GNU-eigenen Projekten sind unter den 2800 Projekten etliche GNU-fremde angesiedelt.

BerliOS [Berli] ist eine Plattform, die hauptsächlich von Mitgliedern des *Fraunhofer Institutes für Offene Kommunikationssysteme (FOKUS)* [FOKUS] betrieben wird. Gemäss eigenen Angaben hat BerliOS sich zum Ziel gesetzt, die unterschiedlichen Interessengruppen im Umfeld der Open Source Software zu unterstützen und dabei eine neutrale Vermittlerfunktion anzubieten. Die Zielgruppen von BerliOS sind einerseits die Entwickler und Anwender von Open Source Software und andererseits kommerzielle Hersteller von Open Source Betriebssystemen und Anwendungen sowie Support-Firmen.

Die bekannteste und mit über 100'000 unterhaltenen Projekten auch grösste Plattform ist *SourceForge* [SrcF]. Sie wird von der *Open Source Technology Group* [OSTG] betrieben und besitzt über eine Million registrierte Benutzer. Die Nutzung der Plattform ist kostenlos. SourceForge bietet eine zentralisierte Projekt-, Versions- und Release-Verwaltung mittels Subversion an. Zur Kommunikation zwischen Entwicklern oder Benutzern stehen den Projekten Mailinglisten und Foren zur Verfügung. So genannte Tracker realisieren das Fehlermelde- und Supportsystem, sowie die Mechanismen zur Überbringung von Benutzerwünschen (*feature requests*).

Mittels webbasierter Werkzeuge können die Projektmitglieder verwaltet und deren Rechte im Projekt konfiguriert werden. Es besteht die Möglichkeit, auf der Projektseite Neuigkeiten zum Projekt zu verkünden und neben dem Quelltext als solches, können vorkompilierte und zur sofortigen Nutzung bereite Programmversionen oder Installationsprogramme der Software zur Verfügung gestellt werden.

Die Projekte können mit Schlüsselwörtern versehen und bestimmten Kategorien zugeordnet werden. So lassen sich Projekte mit der Suchfunktion von SourceForge einfach finden. Ein Projekt wird von SourceForge automatisch mit Statistiken versehen, die Auskunft darüber geben, wie aktiv ein Projekt ist. Für Personen auf der Suche nach einem geeigneten Projekt ist das ein hilfreiches Kriterium für die Wahl eines Projektes. Projekte, die sehr aktiv sind, werden auf der Einstiegsseite von SourceForge prominent beworben.

Eigene Infrastruktur

Wird nicht auf eine vorhandene Plattform zurückgegriffen, so kann ein Projekt auch auf einem eigenen, über das Internet erreichbaren Server unterhalten werden. Auf diesem Server sollten folgende Dienste (mit Beispielanwendungen) laufen:

- Webserver: für die Projektwebseite;
- CVS oder Subversion: Versions- und Release-Verwaltung [CVS];
- Mailman [MLM]: für Mailinglisten;
- phpBB [BB]: Forensoftware.

Um einen derartigen Server einzurichten, muss mit 2 bis 4 Arbeitstagen gerechnet werden, je nach Erfahrung im Umgang mit Internet-Server und je nach Funktionsumfang des Servers. Einmal sauber und korrekt eingerichtet, sollte eine solche Infrastruktur keinen grösseren Aufwand zur Pflege benötigen. Zu den laufend anfallenden Arbeiten gehören z. B. Software-Updates und Backups.

Veröffentlichung

Sobald eine geeignete Open Source Plattform zur Verfügung steht, kann die Veröffentlichung des vorhandenen Projektes unter der gewählten Lizenz angegangen werden. Dazu gehört das Einrichten einer Projekt-Webseite, das allfällige Einspielen des bereits vorhandenen Quelltextes in die Versionsverwaltung, die Erstellung eines ersten Releases und die Einrichtung der Kommunikationskanäle. Der Aufwand für diesen Schritt hängt stark von der verwendeten Open Source Plattform ab. Bei SourceForge nimmt die gesamte Einrichtung etwa einen Arbeitstag in Anspruch.

Da sich die Nutzer eines Projektes grundsätzlich in den beiden Rollen Endbenutzer und Entwickler unterscheiden, sollten für diese verschiedenen Gruppen auch unterschiedliche Informationskanäle eingerichtet werden. Folgende Bezeichnungen haben sich für die Benennung von Mailinglisten resp. von Foren etabliert: *<projectname>-devel und <projectname>-user.*

Nicht nur bei der Benennung von Kommunikationskanälen, sondern auch bei der Strukturierung des Quelltextes, der Unterstützung von Entwicklerwerkzeugen und nicht zuletzt bei der Programmierung des Quelltextes selbst haben sich in der Open Source Community einige Gepflogenheiten durchgesetzt. Manchmal sind die Erwartungen der Open Source Gemeinde derart deutlich, dass die Gemeinde nicht immer sehr *open minded* erscheint. So werden in Foren Neulinge mit scheinbar normalen Fragen mit unfreundlichen Antworten abgekanzelt, sollten sie in der Fragestellung eine der vielen ungeschriebenen Regeln nicht beachtet haben. Die Antworten werden dann oft in einem typischen Abkürzungskauderwelsch gegeben, so dass der verwunderte Neuling nicht mal den Grund der seltsam abweisenden Antwort erahnen kann; oder würden Sie erwarten, dass *RTFM* so viel wie «read the fucking manual» bedeutet? Nicht dass wir selbst auf diese Regeln Wert legen würden, aber ist man am Erfolg des eigenen Open Source Projektes interessiert, lohnt es sich auf einige, zum Teil sogar sinnvolle, Erwartungen der Open Source Gemeinde einzugehen.

Das Open Source Mantra

Als das Open Source Mantra[6] wird oft folgende Regel angesehen: Release early and *release often*. Offenbar wird es in der Open Source Gemeinde wenig geschätzt, wenn ein Projekt fixfertig als Open Source freigegeben wird. Doch noch wichtiger scheint zu sein, dass man regelmässig die Änderungen publik macht und nicht nur grosse Meilensteine veröffentlicht.

Ein Projekt sollte wenn immer möglich auch für die verschiedenen Entwicklungsumgebungen vorbereitet sein. Sehr positiv aufgefasst wird, wenn neben den Microsoft Visual Studio Projektdateien auch die notwendigen GNU Autotools-Dateien (autoconf, automake und libtool) [AuTo] für die Kompilierung des Projektes mit der *GNU Compiler Collection* [GCC] beiliegen.

Damit die Entwicklung auf verschiedenen Plattformen möglichst reibungslos klappt, sollte schon bei der Organisation des Quelltextes auf eine gewisse Verzeichnisstruktur Wert gelegt werden. Folgende Struktur hat sich z. B. für C++-Programme etabliert und wird auch so erwartet:

```
<projektname>-
  |
  +-doc/
  +-include/
  |  +-<projektname>/
  |  |--*.h
  +-src/
  |--*.cpp
```

6 Eine formelhafte Wortfolge, die repetitiv rezitiert wird.

Bei einem Open Source Projekt werden auch bestimmte Dateien erwartet, die unter anderem die Lizenz enthalten (COPYING), Neuigkeiten bereit stellen (NEWS), die Autoren erwähnen (AUTHORS) oder häufig gestellte Fragen beantworten (FAQ).

Über die eingerichteten Kommunikationskanäle wie Mailinglisten und Foren unterhält man sich mit den Interessenten des Projektes. Zu Beginn muss man selbst stark mit Rat und Tat zur Seite stehen, doch je schneller die Zahl der Benutzer steigt, desto rascher ist die Community selbst im Stande, sich mit Hilfestellung zu unterstützen. Es wird zwar stark geschätzt, wenn die Projekt-initiatoren die Kommunikationskanäle weiter verfolgen und wertvollen Rat beisteuern, doch wird das nicht erwartet. Je nach Aktivität in den Mailinglisten und den Foren reicht ein wöchentlicher Besuch in den Mailinglisten und Foren, um das Projekt genügend aktiv zu begleiten.

Wenn das Projekt und die verwendeten Techniken es zulassen, macht es Sinn, ein Projekt möglichst plattformunabhängig zu programmieren. Diese Möglichkeit sollte man auch aus Eigeninteresse nutzen. Das ist nicht nur der grösseren «Mitarbeiterzahl» wegen interessant, sondern auch deswegen, dass das eigene Projekt auf verschiedenen Plattformen betrieben werden kann.

Verwendet man z. B. C++ als Programmiersprache, so lässt diese Sprache die plattformunabhängige Entwicklung grundsätzlich zu. Eine Portierung auf eine andere Betriebsystem-Plattform ist aber nur dann einfach zu bewerkstelligen, wenn die im Projekt verwendeten Programmbibliotheken auch auf der Zielplattform vorhanden sind. Bei C++ Programmen sollte aus diesem Grund möglichst auf die Standardbibliotheken, wie z. B. die *C++ Standard Library* [C++Lib] oder die *Standard Template Library (STL)* [STL], zurückgegriffen werden. Diese beinhalten einige wichtige Datentypen und Funktionen, die in praktisch jedem Programm verwendet werden: Zeichenketten, numerische Berechnungen, Ein- und Ausgabe, Fehlerbehandlung, Laufzeit-Typerkennung, Nationale Besonderheiten, Speicherverwaltung usw.

Überhaupt stellt schon die Wahl der Programmiersprache ein wichtiges Kriterium für die Portabilität dar. C/C++-Programme lassen sich auf fast allen Plattformen kompilieren. Bei Java ist die Plattformunabhängigkeit Konzept. C#-Programme hingegen lassen sich praktisch auf allen Windows-System zuverlässig ausführen. Für Unix-Plattformen kann auf die .NET-kompatible Laufzeitumgebung *Mono* zurückgegriffen werden [Mono].

Fallbeispiel: libPGF

Das vorliegende Fallbeispiel soll den Übergang von proprietärer zu Open Source Software-Entwicklung am Beispiel von *libPGF* [libPGF] illustrieren. *Progressive Graphic File (PGF)* [PGF] ist ein von uns in C++ selbst entwickeltes Bildformat, basierend auf diskreter Wavelettransformation.

Ein Projekt, gerade auch ein Open Source Projekt, braucht einen guten Namen, der nicht bereits von vielen anderen ähnlichen Bedeutungen belegt ist. Bei Dateiformaten fällt die Namenswahl nicht leicht, da unter gewissen Betriebsystemen Dateiformate über eine kurze Dateiendung gekennzeichnet werden. Obwohl gemäss einschlägigen Webseiten [FILExt, FExt] die Dateiendung PGF bereits mehrfach belegt ist («GPS Pathfinder Office Geoid Grid File», «PowerGREP File Selection», «Portfolio Graphics Compressed Bitmap», «Portfolio Graphics image format»), entscheiden wir uns dennoch für das prägnante Kürzel PGF. Den Projektnamen des Open Source Projektes lehnen wir an ähnliche Projekte an, wie zum Beispiel *libpng* [PNG], und wählen libPGF.

Im Zusammenhang mit dem Projektnamen steht eine starke Webpräsenz im Vordergrund. Dabei stellt ein einfaches Auffinden durch Suchmaschinen ein nicht unwesentlicher Erfolgsfaktor für die Bekanntmachung dar. Neben dem eigenen Webauftritt hilft ein Eintrag in Wikipedia [PGFa] für gute Platzierungen bei bekannten Suchmaschinen. Oft verwendete Abkürzungen oder Abkürzungen mit rufschädigenden Assoziationen sollten tunlichst vermieden werden.

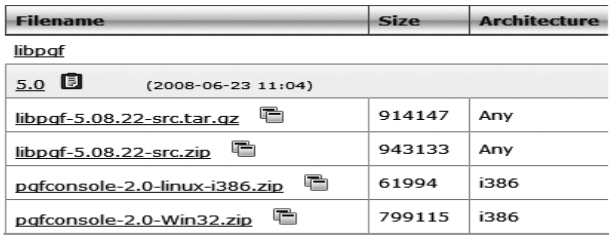
Lizenz

Für Programmbibliotheken ist die Wahl der geeigneten Lizenz entscheidend. Wird eine strenge Lizenz gewählt (z. B. GPL), steht der ideologische Aspekt der Aufrechterhaltung des Open Source Gedankens durch Copyleft im Vordergrund. Wird andererseits eine sehr liberale Lizenz angewendet (z. B. BSD), so steht einer grösstmöglichen Verbreitung nichts im Wege, da auch Hersteller proprietärer Software sich dieser Bibliotheken bedienen können.

Für uns als Entwickler von libPGF spielt der Open Source Gedanke eine wichtige Rolle, doch überwiegt das Interesse einer möglichst grossen Verbreitung. Aus diesem Grund kommt für uns die reine GPL nicht in Frage und LGPL bietet sich als pragmatische Lösung an. Nicht zuletzt da es sich bei libPGF um ein Bibliotheksprojekt handelt, passt die LGPL bestens.

Plattform und Internetpräsenz

Open Source Projekte leben davon, dass der Quelltext und andere Projektdateien der Entwicklergemeinde und den Anwendern zur Verfügung gestellt werden. Grundsätzlich wäre es ausreichend, den Quelltext auf einer eigenen Webseite anzubieten. Typische Abläufe der Softwareentwicklung wären damit aber nicht ausreichend unterstützt. Somit drängt sich die Verwendung



Filename	Size	Architecture
libpgf		
5.0 (2008-06-23 11:04)		
libpgf-5.08.22-src.tar.gz	914147	Any
libpgf-5.08.22-src.zip	943133	Any
pgfconsole-2.0-linux-i386.zip	61994	i386
pgfconsole-2.0-Win32.zip	799115	i386

Abbildung 1: Übersicht der angebotenen Inhalte des Release 5.0.

einer Open Source Plattform auf. Wir haben uns für SourceForge entschieden, da diese Plattform die Erstellung eines Projektes ohne grossen Zusatzaufwand ermöglicht und die mit Abstand bekannteste Plattform von allen ist.

In den nächsten Abschnitten beschreiben wir die von uns vollzogenen Schritte von der Erstellung eines SourceForge-Projektes bis zur Einrichtung einer unabhängigen Projektwebseite.

Projekt erstellen

Die Erstellung eines SourceForge-Projektes ist ziemlich einfach und besteht aus dem Anlegen eines SourceForge-Accounts und der offiziellen Registrierung des Projekts. SourceForge leitet einen sehr gut durch diesen Prozess und macht im Voraus schon darauf aufmerksam, welche Informationen vorhanden sein müssen. So sind unter anderem eine volle Beschreibung des Projektes, die gewünschte Open Source Lizenz und der Name des Projekts nötig. Darüber hinaus können Angaben zu den folgenden Punkten gemacht werden:

- SourceForge Software-Kategorie, in welche das Projekt passt;
- unter welchen Plattformen die Software funktioniert;
- in welcher Programmiersprache die Software geschrieben wurde;
- wie der Entwicklungsstand ist (Alpha, Beta, Version).

Dieser Projektantrag wird dann von Menschenhand kontrolliert und spätestens nach zwei Arbeitstagen erhält man einen Entscheid über Annahme oder Ablehnung des Projektes.

Projekt einrichten

Vorausgesetzt der Antrag ist akzeptiert worden, so steht anschliessend ein Projektplatz [SrcFo] zur Verfügung. Dadurch können die Kommunikationskanäle wie geplant eingerichtet, die Projektangaben detaillierter beschrieben (z. B. Link auf die Projektwebseite), die Versionsverwaltung mit dem Quelltext des aktuellen Entwicklungsstandes eingespeist und vielleicht ein erstes Release bereitgestellt werden.

Im libPGF-Projekt haben wir die folgenden zwei Foren und zwei Mailinglisten als Kommunikationskanäle eingerichtet:

- Das Announce-Forum dient zur Ankündigungen von grösseren Anpassungen unsererseits und ermöglicht das Führen von sachbezogenen Diskussionen.
- Im Discussion-Forum können sich Nutzer und Entwickler zu allen Themen rund um libPGF unterhalten. Das Forum ist auch dafür gedacht, dass sich die Nutzergemeinde hierin selbst hilft.
- Die libpgf-devel-Mailingliste ist für Diskussionen unter Entwicklern der libPGF selbst gedacht.
- Die libpgf-user-Mailingliste ist für Entwickler gedacht, die libPGF für andere Projekte benutzen wollen.

Quelltexte und Releases bereitstellen

In vielen Open Source Projekten werden neben den Quelltexten auch stabile Momentaufnahmen (*Releases*) bereitgestellt. Von diesen stabilen Momentaufnahmen werden oft vorkompilierte Binary-Releases (Binärprogramme) angeboten, um die doch zum Teil mühsamen Kompilationen den Benutzern zu ersparen. Entsprechend diesen Gepflogenheiten bieten wir neben dem aktuellen Entwicklungsstand, stabile Momentaufnahmen des Quelltextes und kompilierte Versionen an. Für die Releases muss ein geeignetes Versionierungsschema her. Die Versionierung bei der libPGF ist so zu lesen: libPGF-5.0 ist der fünfte Haupt-Release der Bibliothek. Eine kleinere (minor) Änderung, würde den Minor-Release libPGF-5.1 ergeben. Eine Fehlerbehebung für diese Version würde das Patchfix-Release libPGF-5.1-1 ergeben.

Um den Einsatz der Bibliothek zu illustrieren, stellen wir der Bibliothek ein prägnantes Demoprogramm im Quelltext und in kompilierter Version zur Seite (siehe auch Abbildung 1). Damit sollten die ersten Bedürfnisse von Anwendern der Bibliothek und jener, die einfach mal einen Blick auf die Fähigkeiten der Bibliothek werfen wollen, befriedigt sein. Die beiden Quelltextpakete (libpgf-5.08.22-src.tar.gz und libpgf-5.08.22-src.zip) unterscheiden sich nur in der Kompressionsart. Entpackt man diese bei sich auf dem Rechner, so erhält man ein Verzeichnis mit den typischen Dateien für ein Open Source Projekt:

```
libpgf/
  +-doc/
  +-include/
  |  +-libpgf/
  +-libpgf.xcodeproj/
  +-src/
  |--AUTHORS
  |--autogen.sh
  |--config.h.in
  |--configure.ac
  |--COPYING
  |--FAQ
  |--INSTALL
  |--libpgf.pc.in
  |--libpgf.spec.in
  |--Makefile.am
  |--NEWS
  |--PGFCodec.vcproj
  |--README
```

Für aktive Entwickler steht der Zugriff auf den aktuellen Quelltext mittels einer Versionsverwaltung zur Verfügung. Bei SourceForge kommt Subversion hierfür zum Einsatz. Der Lesezugriff auf die Dateien in der Versionsverwaltung ist aber auch jedem anderen Benutzer gestattet. Einen schreibenden Zugriff bekommen aber nur diejenigen SourceForge-Benutzer, denen wir als Projektverantwortliche Zugriff gestatten. Der Zugriff auf die Versionsverwaltung Subversion geschieht mittels dem Protokoll https und kann praktisch in jeder Entwicklungsumgebung komfortabel eingebunden werden[7].

Projektwebseite

Eine Projektwebseite dient der Verbreitung von allgemeinen Informationen zum Projekt. Es wird beschrieben, worum es sich beim Projekt handelt, welche Features angeboten werden, welche Einsatzgebiete geeignet sind, wer dahinter steht und welche Hintergrundinformationen zur Verfügung stehen. Nicht zuletzt dient eine solche Seite Marketingzwecken. Marketingtechnisch ist eine Projektwebseite, die unter einem eigenen, aussagekräftigen Domainnamen erreichbar ist, wertvoller als eine Seite, die unter einer Subdomain oder in einem Unterverzeichnis zu finden ist.

Ein Projektplatz bei SourceForge ist hauptsächlich als Kollaborationswerkzeug für die Entwicklung gedacht. Trotzdem bietet SourceForge pro Projekt 100 MByte Speicherplatz, PHP und eine MySQL-Datenbank an, um unter einer Subdomain von SourceForge.net eine Projektwebseite zu betreiben. Dabei setzt sich die URL folgendermassen zusammen: http://<projektname>.sourceforge.net.

Trotz dieser Möglichkeit haben wir uns für eine eigene Domain entschieden und betreiben die Projektwebseite bei einem selbst gewählten Hoster[8]. Dort bieten wir weitere Informationen zur libPGF an und verweisen für die Downloads und die Supportkanäle auf die entsprechenden SourceForge-Projektseiten. Vom Sourceforge-Projektplatz wiederum verweisen wir zur eigenen Projektwebseite www.libpgf.org.

Fazit und Schlussbemerkungen

Unsere bisherigen Erfahrungen mit dem Open Source Projekt libPGF sind durchwegs gut. Gleich zu Beginn haben wir zwei sehr interessierte Personen über die Kommunikationskanäle kennen gelernt. Während die erste Person uns bei der Erstellung der Releases geholfen hat (GNU Autotools und Vorschlag wie die Verzeichnisstruktur geändert werden sollte), zeigte die andere Person primär Interesse am Bildformat. Auf uns nicht bekanntem Weg fand das Bildformat denn auch schon Einzug in die Open Source Game-Engine OGRE [OGRE]. Das ist natürlich eine sehr schöne Sache.

Der weitere Aufwand für den Unterhalt des Projektes geht gegen null. Dann und wann sehen wir wieder mal in den Foren nach, ob da plötzlich eine rege Diskussion statt findet (was nicht der Fall ist) und bei den eingerichteten Mailinglisten habe wir uns natürlich auch angemeldet. Sendet jemand eine Meldung an diese Listen, so erhalten wir diese per E-Mail und können darauf antworten.

Genau zwei Jahre nach Veröffentlichung des Projektes sind der Quelltext der Bibliothek libPGF und das Demoprogramm gesamthaft 1180mal bezogen worden. Diese Zahlen sind nicht berauschend, was sicherlich auch damit zusammenhängt, dass sich unsere Aktivitäten auf ein Minimum beschränkten. Gerade was das Marketing anbelangt, hätten wir einiges mehr tun können. Zum Beispiel wäre ein Pressecommuniqué an die führenden Online-News-Dienste und technischen Verlage der Aufmerksamkeit dienlich gewesen.

7 Für Microsoft Visual Studio existiert z. B. Ankhsvn: http://ankhsvn.tigris.org/

8 In unserem Fall ist die Projektwebseite bei METANET GmbH, Switzerland, gehostet.

Referenzen

[Apache] Apache Software Lizenztext: http://www.apache.org/licenses/LICENSE-2.0.html

[AuTo] GNU Software: http://www.gnu.org/software/

[BB] PHP Bulletin Board: http://www.phpbb.com

[Berli] BerliOS: http://www.berlios.de

[BSD] BSD-Lizenztext: http://www.opensource.org/licenses/bsd-license.php

[C++Lib] GNU C++ Standard Library: http://gcc.gnu.org/onlinedocs/libstdc++/documentation.html

[CHLR] Landesrechte: http://www.admin.ch/ch/d/sr/index.html

[CVS] Concurrent Versions System: http://www.nongnu.org/cvs/

[FExt] File Extensions: http://www.file-extensions.org

[FILExt] FILExt: http://filext.com

[FOKUS] FOKUS Institut: http://www.fokus.fraunhofer.de/home/

[FSF] Free Software Foundation: http://www.fsf.org

[GCC] GCC, the GNU Compiler Collection: http://www.gnu.org/software/gcc/

[GNU] GNU Free Documentation License: http://de.wikipedia.org/wiki/Wikipedia

[GPL] GNU General Public License: http://www.gnu.org/copyleft/gpl.html

[GPLv3] Offizieller GPLv3-Text: http://www.gnu.org/licenses/gpl.html

[IFROSS] Institut für Rechtsfragen der Freien und Open Source Software: http://www.ifross.de/ifross_html/lizenzcenter.html

[LGPL] LGPL-Text: http://www.gnu.org/licenses/lgpl.html

[libPGF] libPGF Open Source Image Codec: http://www.libpgf.org/

[MLM] GNU Mailing List Manager: http://www.gnu.org/software/mailman/

[Mono] Mono Project: http://www.mono-project.com/Main_Page

[MPL] MPL-Text: http://www.mozilla.org/MPL/MPL-1.1.html

[OGRE] Object-Oriented Graphics Rendering Engine; OGRE: http://www.ogre3d.org

[OSI] Open Source Initiative: http://www.opensource.org

[OSIa] Licenses by Name: http://opensource.org/licenses/alphabetical

[OSL] Open-Source-Lizenzen: http://openfacts.berlios.de/index.phtml?title=Open-Source-Lizenzen

[OSTG] OSTG: http://www.ostg.org

[PGF] xeraina PGF: http://www.xeraina.ch/pgf/

[PGFa] Progressive Graphics File: http://de.wikipedia.org/wiki/Progressive_Graphics_File

[PNG] Portable Network Graphics, lizenziert unter der zlib/libpng-Lizenz: http://www.libpng.org

[QG] GPLv3 Quick-Guide: http://www.gnu.org/licenses/quick-guide-gplv3.html

[Sav] Savannah: http://savannah.gnu.org

[SrcF] SourceForge: http://www.sourceforge.net

[SrcFo] SourceForge libPGF: https://sourceforge.net/projects/libpgf/

[STL] STL Programmer's Guide: http://www.sgi.com/tech/stl/

[SVN] Subversion: http://subversion.tigris.org/

[URG] Bundesgesetz über das Urheberrecht und verwandte Schutzrechte: http://www.admin.ch/ch/d/sr/c231_1.html

[Wiki] Wikipedia – Die freie Enzyklopädie: http://de.wikipedia.org