Deutsche Stiftung für Recht und Informatik

Open Source, Open Core, Multi Licensing – die eigene Software richtig als Open Source auslizenzieren

Dr. Hendrik Schöttle
Osborne Clarke

Herbstakademie 2022

1. MITWIRKUNG AN OSS-PROJEKTEN

Mitwirkung an OSS-Projekten

- Warum mitwirken?
- Hoher Aufwand des Forking bei Implementierung eigener Änderungen in den offiziellen Branch: Modifikationen in interner Kopie des offiziellen Branches müssen bei allen neuen Versionen des offiziellen Branches erneut implementiert werden
- Eine einmal vorgenommene Änderung führt damit zu wiederkehrenden Folgeänderungen in den jeweils folgenden offiziellen Versionen

Mitwirkung an OSS-Projekten

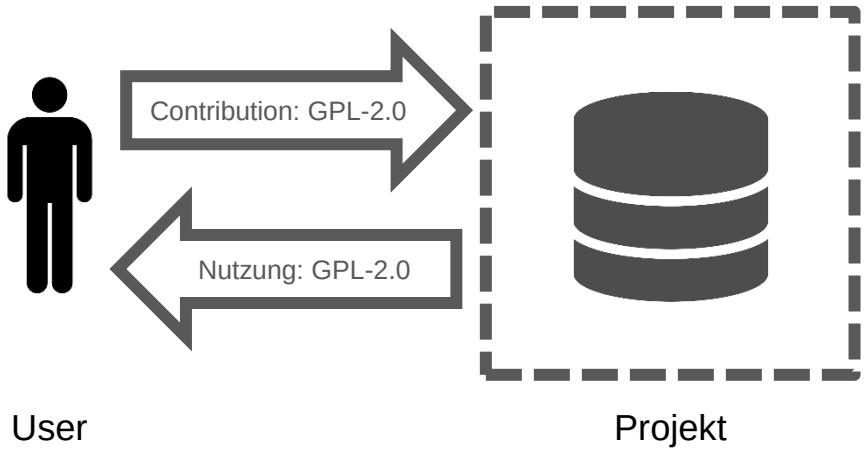
Kostenminimierung

- Eine Community kann nur dann bei der Minimierung von Kosten helfen, wenn sie unabhängig ist
- Unternehmen, die von Community-Arbeit profitieren, sollten den Bestand der Community und Code-Qualität sicherstellen
- Mitarbeiter brauchen klare Anweisungen, welchen Code sie zurück an die Community geben dürfen. Abzuwägen sind:
  - der Schutz von IP-Rechten des Unternehmens
  - eine praktikable Einstellung zu Beiträgen an die Community – interne Freigabeprozesse für jede einzelne Line of Code sind meist nicht praktikabel

Contributions | Offenes Projekt ohne CLA

- Contribution und Nutzung möglich, sie erfolgen unter derselben Lizenz (Inbound = Outbound)
- Vorteile:
  - einheitliches Rechtsregime für alle
  - Kein zentrales Rechtsmanagement erforderlich
- Nachteil:
- weder Umlizenzierung noch duale Lizenzierung für Projektinhaber möglich
- Lizenzwahl gilt, falls es keine Upgrade-Klauseln gibt, für die Ewigkeit

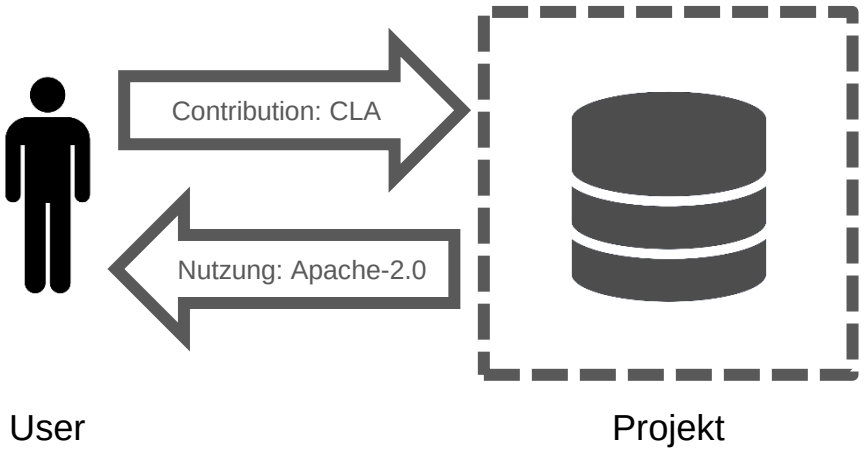
Beispiel: Linux Kernel



Contributions | Offenes Projekt mit CLA

- Contribution und Nutzung möglich, sie erfolgen unter unterschiedlichen Lizenzen (Inbound ≠ Outbound)
- In der Regel weitere Rechtseinräumung bei Contributions als umgekehrt
- **Vorteil**: Umlizenzierung und kommerzielle Nutzung bleiben dem Projektinhaber möglich
- **Nachteil**: zentrales Rechtsmanagement erforderlich
- Aber: Forking ebenfalls möglich, dann aber nur unter der Nutzungs-Lizenz

Beispiel: Apache Foundation



2. DAS EIGENE OSS-PROJEKT

Das eigene OSS-Projekt

Grundüberlegungen

- Endgültigkeit der einmal getroffenen Entscheidung:
  - Aufgrund Reziprozität der Open-Source-Lizenzen kann eine einmal auslizenzierte Software unbegrenzt weiterverbreitet werden, auch wenn der Lizenzgeber sie von seiner Plattform löscht und sie selbst nicht mehr verbreitet
  - Ein „Zurückholen“ von OSS ist nicht möglich
- Umlizenzierung später zwar grundsätzlich möglich (z.B. spätere Kommerzialisierung). Aber:
  - ältere Versionen können immer unter der ursprünglichen Version genutzt werden
  - Einsatz älterer Versionen jedoch meist aus Sicherheitsgründen nicht ratsam

Das eigene OSS-Projekt

Grundüberlegungen

- Veröffentlichung des Source Code als OSS nicht per se „schlecht“:
  - Verfügbarkeit des Source Code bedeutet nicht zwingend Unabhängigkeit vom Hersteller. In der Regel hat der Nutzer keine Kapazitäten, den Source Code zu verstehen und selbst (oder durch Dritte) weiterzuentwickeln oder zu pflegen
  - OSS kann auch als Türöffner oder zum leichteren Markteintritt genutzt werden, später dann Kommerzialisierung des ursprünglich offenen Produkts
  - Aber: nur eine Minderheit der Nutzer können in zahlende Kunden konvertiert werden. Auch möglich, dass die Community oder Dritte die letzte freie Version forken (siehe z.B. https://www.heise.de/news/OpenSearch-1-0-Amazons-Elasticsearch-Fork-ist-fertig-6136614.html)

Das eigene OSS-Projekt

Grundüberlegungen

- Sicherheitsaspekte prüfen! Software sollte bei Veröffentlichung state-of-the-art sein, wenn in sicherheitsrelevanten Bereichen eingesetzt
- Security by Obscurity funktioniert dann nicht mehr (war sowieso nie eine gute Idee, allein darauf zu setzen)
- OSS Compliance muss natürlich auch im eigenen Projekt gegeben sein – eine vollständige SBOM und alle anderen Pflichtinformationen sind dann Pflicht!

Das eigene OSS-Projekt

Grundüberlegungen

- OSS besonders bei Commodities sinnvoll (Software, die ohnehin jedes Unternehmen braucht und die nicht differenzierend ist, z.B. Betriebssystem, Standardfunktionen etc. Beispiel: https://www.openautoalliance.net/)
- Andere Motivation: Einsammeln von Contributions aus Community, um eigene Produkte voranzutreiben.
- Zahlreiche andere Arten des Einsatzes von OSS für eigene Projekte denkbar, etwa als Inner Source. Siehe dazu auch den Leitfaden Open Source des BITKOM, https://www.bitkom.org/Bitkom/Publikationen/Open-Source-Leitfaden-Praxisempfehlungen-fuer-Open-Source-Software-Version-30

3. WELCHE OSS-LIZENZ NEHMEN?

Welche OSS-Lizenz nehmen?

- Welche Lizenz soll verwendet werden?
- Verschiedene Aspekte müssen beachtet werden:
  - Starker Copyleft-Effekt (z.B GPL) oder nicht?
  - Permissive (z.B. MIT oder BSD) oder nicht?
  - Kompatibilität der Lizenzen (z.B. Integration in vorhandene GPL-Komponenten)
- Die Auswahl der Lizenz hängt besonders davon ab, welche Ziele verfolgt werden
- Rad nicht neu erfinden. Keine Lizenz ist perfekt. Aber Verwendung bekannter Lizenzen sorgt für höhere Akzeptanz im Markt bzw. in Communities. Englischsprachige Lizenzen bevorzugen, da schnell eine größere Community erreicht werden kann. Exotisch und wenig bekannte Lizenzen vermeiden

Welche OSS-Lizenz nehmen?

- Was will ich erreichen? Beispiele für Ziele und entsprechend passende Lizenzen:
  - Verbesserungen und Updates durch die OSS-Community = GPL-2.0?
    Nicht zwingend! Gut gemanagtes Projekt unter der BSD/MIT ist mehr wert, als unkoordiniertes Einsammeln von GPL Contributions
  - Schutz vor SaaS = AGPL, SSPL
  - Schutz vor Patentverletzungsklagen = GPL-3.0 oder andere Lizenz mit Patent Retaliation Clause
    - Aber: viele Unternehmen vermeiden GPL-3.0
    - Gefahr des Patentleft-Effekts
    - GPL-3.0 in vielen Szenarien nicht einsetzbar (z.B. DRM-geschützte „User Products“ wie iPhone Apps)
- Jedes Open-Source-Projekt, welches unter eine bekannte OSS-Lizenz gestellt wird, vermeidet es, die Zahl der verwendeten Lizenzen im Feld – und damit den Prüfungsaufwand im Rahmen der Open-Source-Compliance – noch weiter zu erhöhen

Welche OSS-Lizenz nehmen?

Open Core

- Kernbestandteil der Software als OSS frei verfügbar
- Bestimmte Zusatzfeatures und Add-Ons kostenpflichtig unter einer kommerziellen Lizenz
- In Open Source Communities teilweise in Verruf geraten. Grund unter anderem unklare Lizenzgestaltung. Nutzer tappt schnell in die Falle der Unterlizenzierung und muss fehlende Nutzungsrechte nacherwerben
- Spannungsfeld AGB-Recht und Urheberrecht
  - Unbedingt ganz klar definieren, welche Komponenten wie lizenziert sind
  - Ansonsten: beabsichtigtes Verbot bestimmter Nutzungsarten greift schlimmstenfalls nicht, wenn es unklar formuliert ist
  - Vgl. OLG Köln, Urt. v. 31.10 2014 - I-6 U 60/14, Rn. 81

Welche OSS-Lizenz nehmen?

Open Core

- Lizenzgeber muss sicherstellen, dass er an kommerziell lizenzierten Teilen ausreichende Nutzungsrechte besitzt
- Bei Weiterentwicklung eines Drittprojekts können nur solche Komponenten kommerziell auslizenziert werden, welche der Lizenzgeber vollständig selbst entwickelt hat
- Aufpassen bei Copyleft-Lizenzen: Wird an z.B. GPL-lizenzierte Software ein kommerzielles Plugin angedockt, muss der Lizenzgeber möglicherweise mit sogenannten Linking Exceptions den Copyleft-Effekt einfangen

Welche OSS-Lizenz nehmen?

Open Core/Multi Licensing

- Achtung: selbst strenge OSS-Lizenzen wie AGPL stehen nicht zwingend kommerziellen Einsatzszenarien im Weg. Ergebnis: kommerzielle Lizenz nicht immer erforderlich.
- Insbesondere im Cloud-Bereich wurde zuletzt häufiger etwa von der AGPL oder der GPL auf noch strengere Source-Available-Lizenzen wie die SSPL gewechselt, weil selbst unter der recht strengen AGPL-3.0 bei einigen Einsatzszenarien der Software der Copyleft-Effekt nicht ausgelöst wurde – so dass auch keine Notwendigkeit zum Erwerb einer kommerziellen Lizenz bestand.
- Lizenzgeber, der auf Multi Licensing setzt, sollte sich vorher genau prüfen, welche Einsatzszenarien Gegenstand einer kostenpflichtigen Lizenzierung sein sollen, um die dafür passende (Open-Source-)Lizenz zu wählen.

Kontakt

Dr. Hendrik Schöttle
Partner, Fachanwalt für IT-Recht
Germany

+49 89 5434 8046
hendrik.schoettle@osborneclarke.com

„Im Bereich Open Source ein Spitzenname“

Wettbewerber, JUVE-Handbuch 2021/2022

- Dr. Hendrik Schöttle berät im IT- und Datenschutzrecht.
- Hendrik Schöttle wurde in den letzten Jahren wiederholt sowohl vom Handelsblatt und von Best Lawyers als auch von der Wirtschaftswoche und vom Kanzleimonitor als einer der besten Anwälte bzw. als mehrfach empfohlener Anwalt im IT-Recht genannt. Laut JUVE-Handbuch 2021/2022 ist er „im Bereich Open Source ein Spitzenname“. Das Kanzleihandbuch Legal 500 Deutschland empfiehlt ihn, weil er durch „sehr gute IT-Kenntnisse besticht, auch wenn es sich um exotische Fragen handelt“ und durch ein „sehr schnelles Verständnis technischer Details“.
- Er hat langjährige Erfahrung bei der Beratung, Vertragsgestaltung und Verhandlung von komplexen IT-Projekten. Seine Schwerpunkte sind IoT, Digitalisierung und Cloud Computing. Er berät zu Software-Lizenzmodellen, insbesondere zu Open-Source-Software, und im Datenschutzrecht. Zu seinen Mandanten gehören international tätige Technologiekonzerne sowie namhafte IT- und E-Business-Unternehmen.

- Hendrik Schöttle arbeitet seit 2005 als Rechtsanwalt, seit 2007 im Münchner Büro von Osborne Clarke. Er war mehrfach im Rahmen von Secondments in Rechtsabteilungen von IT-Unternehmen tätig. Zudem hat er mehrere Jahre als Software-Entwickler am Institut für Rechtsinformatik der Universität des Saarlandes gearbeitet. Seine praktische Erfahrung und sein technisches Know-how kommen seinen Mandanten bei der technologienahen Beratung zugute.
- Er ist Autor zahlreicher Veröffentlichungen, Mitautor mehrerer Handbücher und Kommentare, unter anderem des Beck’schen Handbuchs IT- und Datenschutzrecht und des juris Praxiskommentars zum BGB.
- Hendrik Schöttle ist Dozent der Deutschen Anwaltakademie für den Fachanwaltslehrgang IT-Recht und hält regelmäßig Vorträge zu Themen des IT-Rechts.
- Er ist Mitglied im Vorstand des Arbeitskreises Open Source des BITKOM, Mitglied des Ausschusses Datenschutzrecht der Bundesrechtsanwaltskammer (BRAK), der Arbeitsgemeinschaft Informationstechnologie im Deutschen Anwaltverein (DAV) und der Deutschen Gesellschaft für Recht und Informatik (DGRI).