Geschäftsrisiken und Governance von Open-Source in Softwareprodukten

Autoren: Martin Helmreich, Dirk Riehle

Stichwörter: Open-Source-Komponenten, Open-Source-Governance, Geistiges Eigentum, Code-Scanner, Softwareprodukte

Kurztitel: Risiken von Open-Source-Komponenten

Referenz: In *Praxis der Wirtschaftsinformatik* (HMD 283) 49. Jahrgang, Februar 2012

Abstract

In fast jedem Softwareprodukt, auch in großer Standardsoftware, sind heute Open-Source-Komponenten enthalten. Die Hersteller dieser Software müssen die Geschäftsrisiken, die mit der Integration von Open-Source-Software in kommerzielle Produkte verbunden sind, verstehen und vernünftig managen. Dieser Artikel zeigt ein Modell verschiedener rechtlicher, technischer und sozialer Risiken auf, die durch unkontrollierten Einsatz von Open-Source-Software entstehen und erläutert ausgewählte Erfolgsmethoden der Open-Source-Governance, die von führenden Firmen angewandt werden. Das Modell ist das Analyseergebnis von fünf mit großen deutschen Softwareherstellern geführten Interviews sowie weiterer Literaturrecherche.

Inhaltsübersicht

1. Open-Source-Komponenten in kommerziellen Produkten

Was haben Google, Apple, Microsoft, SAP, Samsung, BMW, Citroën gemeinsam? Alle integrieren Open-Source-Software in ihre kommerziellen Produkte. Kaum jemand, der Software für eine übliche Plattform entwickelt, kommt mit Open-Source-Software nicht in Berührung. Die Anwendungsfälle sind dabei so unterschiedlich wie die Firmen, die die Komponenten in ihre Produkte integrieren. Angefangen von der Verwendung einer einzelnen Bibliothek über die Nutzung einer Auswahl von Standardkomponenten bis hin zu großen Projekten wie Android oder Industrie-Konsortien wie Genivi reicht die Vielfalt an Einsatzvarianten.

Android ist auch für nicht-Softwareentwickler und nicht-Open-Source-Experten ein fester Begriff geworden. Abgesehen von Einsatzgebieten wie Tablet-PCs oder Netbooks verzeichnete Android einen signifikanten Gewinn an Marktanteilen: Laut comScore lag dieser im Oktober 2009 in den USA noch bei 2,8% und hat sich bis heute (Stand Juli 2011) auf 41,8% knapp verfünfzehnfacht.

Unter dem Namen Genivi haben führende Automobilhersteller und deren Zulieferer im März 2009 ein Konsortium ins Leben gerufen, das eine herstellerübergreifende Plattform für Infotainment-Systeme im Auto (In-Vehicle Infotainment) entwickeln soll. Mittlerweile sind mehr als 100 Firmen Mitglied bei Genivi und damit hat das Thema Open-Source eine große Reichweite in der Automobilindustrie.

Warum nutzen so viele Firmen Open-Source-Software in ihren kommerziellen Produkten?

- Open Source hilft Kosten zu senken. Da Open-Source-Komponenten per Definition kostenfrei erhältlich sind, können durch ihren Einsatz teilweise erhebliche Lizenzkosten eingespart werden.
- Open Source hilft, die Abhängigkeit von Herstellern und deren (Preis-)Politik zu reduzieren.
- Open Source hilft, sich auf wettbewerbsdifferenzierende Funktionalität zu konzentrieren. Da durch den Einsatz von Open-Source-Komponenten viele Standardfunktionalitäten bereitgestellt werden können, kann sich die Entwicklungsmannschaft auf wettbewerbsdifferenzierende Merkmale der Software konzentrieren. Dadurch wird einerseits nicht unnötig „das Rad neu erfunden“ und der x-te XML-Parser implementiert und andererseits kann das Team viel intensiver spezifisches Fachwissen aufbauen und weiterentwickeln.

2. Methodisches Vorgehen

In den Jahren 2010 und 2011 haben wir ein Projekt durchgeführt, dessen Ziel es war, die Konsequenzen des Einsatzes von Open-Source-Software in Closed-Source-Softwareprodukten zu verstehen [Helmreich 2011]. Dieser Artikel stellt wesentliche Ergebnisse dieser Arbeit dar. Insbesondere beschreibt er ein Modell der sich für Hersteller ergebenden Geschäftsrisiken und stellt einige ausgewählte Erfolgsmethoden („best practices“) dar, mit denen Hersteller diesen Geschäftsrisiken proaktiv begegnen können.

Unser methodisches Vorgehen war das folgende: Zuerst haben wir die existierende Literatur ausgewertet, um uns ein initiales Bild zu machen. Im Anschluss daran haben wir 5 explorative Interviews mit großen deutschen Softwareherstellern geführt, um dieses Bild zu verfeinern. Im Sinne der Grounded Theory sind wir ohne vorgefasste Meinung oder Hypothesen an diese Arbeit herangegangen [Strauss et al. 1996]. Das Ergebnis der Literaturanalyse zusammen mit den Interviewanalysen ist die Ableitung wiederkehrender Begriffe (als Codes), deren Klassifizierung und Kategorienbildung, und schlussendlich die Bestimmung der relevanten abstrakten Begriffe, welche das Hauptergebnis dieser Arbeit ausmachen. Diese Begriffe sind die in diesem Artikel benannten und geschilderten Geschäftsrisiken und Erfolgsmethoden. Die dargestellten Arbeitsergebnisse dieser explorativen Forschung sind somit bisher nicht in signifikantem Umfang empirisch validiert worden, wir glauben aber, dass aufgrund der Kongruenz der erarbeiteten Konzepte und ihrer Beziehungen eine zukünftige quantitative Validierung diese im Wesentlichen bestätigen wird.

Der resultierende Forschungsbeitrag dieses Artikels umfasst:

- Das erste uns bekannte Modell von Geschäftsrisiken, welchen sich Softwareproduktfirmen ausgesetzt sehen, wenn sie Open-Source-Software einsetzen.
- Die exemplarische Darstellung ausgewählter Erfolgsmethoden der Open-Source-Governance, welche Softwarefirmen einsetzen, um diesen Risiken zu begegnen.

Untersuchungsgegenstand war stets die Integration von Open-Source-Komponenten oder –Codefragmenten in kommerzielle Produkte, die nicht unter einer Open-Source-Lizenz zur Verfügung stehen (d.h. i.d.R. verkauft werden). Explizit nicht betrachtet wird der Einsatz von Anwendersoftware (z.B. Firefox, Open/LibreOffice, Eclipse, etc.) im Büro. Diese Unterscheidung ist essentiell, da die meisten (wenn nicht alle) Open-Source-Lizenzen lediglich die Verteilung der Software mit Lizenzverpflichtungen belegen, während die bloße Verwendung ohne Auflagen (oft sogar ohne die Lizenz zu akzeptieren) erfolgen kann.

3. Grundlagen zum geistigen Eigentum

Die meisten Implikationen, die durch die Verwendung von Open-Source-Komponenten in kommerziellen Produkten entstehen, hängen mit geistigem Eigentum zusammen. Drei verschiedene Ausschlussrechte schützen das geistige Eigentum: Das Urheberrecht, das Patentrecht und das Markenrecht. Diese drei haben unterschiedlich starke Bedeutung, wenn es um den Einsatz von Open-Source-Komponenten geht.

Urheberrecht

Durch das Urheberrecht wird jedes Werk intellektueller Arbeit, gleichgültig ob Text, Musik, Film, Bilder oder irgendein anderes Werk, in seiner Ausdrucksform geschützt. Dabei bezieht sich der Schutz ausschließlich auf die Art und Weise der Darstellung und ist unabhängig von den eigentlichen Inhalten. So ist zum Beispiel bei einer Implementierung eines Algorithmus‘ genau die gewählte Form geschützt, nicht jedoch der Algorithmus selbst. Triviale Code-Teile, die jeder Programmierer auf die gleiche oder ähnliche Art und Weise verwenden würde, sind nicht geschützt. Das Urheberrecht gewährt ausschließlich dem Urheber das Recht das Werk zu kopieren, zu verbreiten, zu verändern oder öffentlich vorzuführen. Daher muss der Urheber (Autor) anderen explizit das Recht einräumen, sein Werk zu verbreiten und/ oder zu verändern, damit es von Dritten in deren Produkte integriert werden kann. Dies geschieht in der Regel durch Open-Source-Lizenzen (siehe unten). Die von 164 Nationen unterzeichnete „Berner Übereinkunft zum Schutz von Werken der Literatur und Kunst“ stellt sicher, dass das Urheberrecht in weiten Teilen der Welt einheitlich und länderübergreifend geregelt und durchgesetzt wird. So besteht der Schutz automatisch bei Entstehung eines Werkes ohne Zutun des Urhebers (es ist also auch kein „© Autor“ oder „Copyright Autor“ notwendig, um das Werk urheberrechtlich zu schützen) und besteht in den meisten Fällen für mindestens 50 Jahre. Nach dieser Zeit wird das Werk gemeinfrei (engl. „public domain“) und ist nicht länger geschützt. Während es in den USA beispielsweise möglich ist, ein Werk direkt als gemeinfrei zu erklären und damit auf das Urheberrecht zu verzichten, ist ein derartiger Verzicht in Deutschland nicht möglich. Hierzulande können nur Verfügungen über sämtliche Nutzungsrechte getroffen werden, zum Beispiel kann jedermann das Recht eingeräumt werden, das Werk zu kopieren, zu verbreiten, zu verändern und öffentlich vorzuführen. Eine entscheidende Frage in diesem Zusammenhang ist, was alles als Veränderung (Bearbeitung, abgeleitetes Werk, engl. „derivative work“) zu sehen ist, da eben für jede Veränderung die Zustimmung des Autors notwendig ist. Viele Lizenzen legen fest, was darunter zu verstehen ist.

Patente

Patente schützen Erfindungen. Ein Patent gewährt dem Inhaber das Recht, andere von der Herstellung, Verwendung, dem Angebot zum Verkauf oder dem Verkauf einer Erfindung auszuschließen. Niemand darf ohne die Zustimmung des Patentinhabers irgendetwas tun, das auf der geschützten Erfindung basiert. Um eine Erfindung patentieren lassen zu können, müssen drei Bedingungen erfüllt sein:

1. Die Erfindung darf nicht Stand der Technik sein. Da alle bisherigen Veröffentlichungen Teil des Stands der Technik sind, stellt die Prüfung, ob eine Erfindung nicht doch schon Stand der Technik ist, den aufwendigsten Teil einer Patenterteilung dar.
2. Die Erfindung muss Erfindungshöhe haben. Ein Experte auf dem entsprechenden Gebiet darf nicht ohne weiteres auf dieselbe Idee kommen dürfen.

3. Die Erfindung muss kommerziell nutzbar sein. Das ist fast immer der Fall.

Ein spezieller Fall sind Softwarepatente. Während in Deutschland Software per se nicht patentierbar ist, kann Software in anderen Ländern ohne weiteres patentiert werden. Auch in Deutschland kann Software indirekt patentiert werden.

Markenrechte

Markenzeichen schützen Zeichen (Logos, Phrasen und Kombinationen daraus) und sind an Produkte gebunden. Sie können registriert werden und werden in einem Register veröffentlicht.

Softwarelizenzen

Während durch Markenzeichen keine besondere Situation für Open-Source-Software entsteht, ist es notwendig urheberrechtliche Regelungen zu treffen und sinnvoll patentrechtliche Aspekte zu regeln. Urheberrechtliche Erklärungen sind Grundvoraussetzung dafür, dass Open-Source „funktionieren“ kann. Denn nur wenn der Autor einer Komponente anderen Rechte daran einräumt, darf sie von Dritten verwendet werden. Patentrechtliche Regelungen sind vor allem im kommerziellen Umfeld interessant, da ein Unternehmen, das Software eines Autors einsetzt, die vom Autor patentierte Bestandteile enthält, eine Patentlizenz benötigt, um diese Software nutzen zu dürfen.

Open-Source-Lizenzen bieten einen (halbwegs) einheitlichen Rahmen für diese Regelungen und ersparen dem Urheber eigene (womöglich nicht rechtswirksame) Formulierungen. Dabei sind drei Kategorien von Lizenzen zu unterscheiden:

1. Permissive Lizenzen. Diese Lizenzen gewähren dem Lizenznehmer (dem Nutzer) in der Regel umfassende Nutzungsrechte einschließlich dem Recht, die Software unter anderer Lizenz an Dritte weiterzugeben, ohne oder nur mit geringen Einschränkungen.
2. Copyleft-Lizenzen gewähren dem Nutzer umfassenden Nutzungsrechte unter der Bedingung, dass er die (veränderte) Software ausschließlich unter derselben (oder ggf. einer inhaltlich kompatiblen) Lizenz an Dritte weitergibt.
3. Virale Copyleft-Lizenzen weiten das Copyleft auf alle abgeleiteten Werke aus. Somit muss jede Software, die ein abgeleitetes Werk einer derart lizenzierten Komponente darstellt, ebenfalls unter der Open-Source-Lizenz weitergeben werden.

Infobox Copyleft

Die Idee des Copylefts ist, das Copyright (das Urheberrecht) zu nutzen, um dieses „umzukehren“. Aus dem Copyright, das ermöglicht, andere auszuschließen, wird das Copyleft, das es unmöglich macht, andere auszuschließen. Dabei vergibt der Urheber umfassende Nutzungsrechte, die jedem eine Bearbeitung und Verbreitung der Software ermöglichen, allerdings unter der Bedingung, dass jeder der diese Rechte nutzt, sie auch an Dritte, an die er die Software verteilt, weitergibt. So wird sichergestellt, dass Software, die einmal unter einer Open-Source-Lizenz frei verfügbar war, immer frei verfügbar bleiben wird.

4. Identifizierte Geschäftsrisiken

Dieser Abschnitt stellt ein Modell von Geschäftsrisiken (siehe Abb. 1) dar, welche sich aus Interviews und Literaturrecherche ergeben haben. Die verwendete Literatur war [Badenhope 2010; Black Duck Software 2010; Hammond 2009; OpenLogic 2010a; OpenLogic 2010b; OpenBRR 2005; Radcliffe 2010; Sutor 2010]. Die Interviews wurden mit 5 großen deutschen Softwareentwicklungsfirmen (>1.000 Angestellte) geführt; die Unternehmen wollten aber anonym bleiben und werden deswegen nicht namentlich genannt.

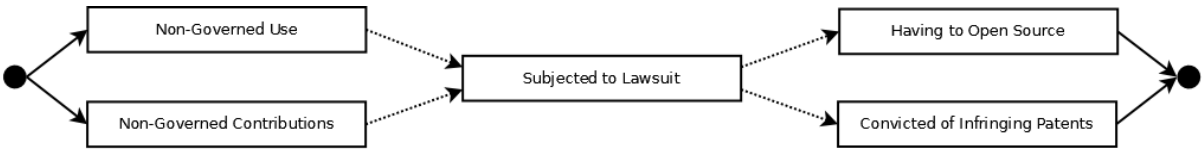

Abb. 1. Darstellung möglicher Geschäftsrisiken durch Open Source

4.1. Unkontrollierter und ungeregelter Einsatz von Open-Source-Komponenten

Das initiale Risiko stellt der unkontrollierte und ungeregelte Einsatz von Open-Source-Software dar. Ist es einem Entwickler erlaubt resp. wird nicht kontrolliert, dass/ob er beliebige Open-Source-Komponenten in ein Produkt einbaut, so folgen ggf. unerwünschte Konsequenzen, wie bereits im Abschnitt zum geistigem Eigentum geschildert.

- Es ist nicht möglich eine genaue Auflistung der verwendeten Open-Source-Komponenten und deren Lizenzen zu erstellen. Eine solche „Stückliste“ wird aber von vielen Kunden gefordert. An diese Kunden kann das Produkt nicht verkauft werden.
- Ein ähnliches Problem ergibt sich bei möglichen Unternehmenszusammenschlüssen und -übernahmen. Der Wert eines Unternehmens lässt sich nicht sicher ermitteln, wenn keine verlässlichen Aussagen vorliegen, welcher Anteil des Codes gar nicht geistiges Eigentum des Unternehmens ist. Vor allem lässt sich nicht bestimmen, ob die Lizenzbedingungen des Open-Source-Codes eingehalten wurden oder ob daraus eine weitere Wertminderung entsteht.
- Das fehlende Wissen über die rechtlichen Verpflichtungen, die im eigenen Source-Code durch die Open-Source-Komponenten enthalten sind, erhöht das Risiko in einen Rechtsstreit verwickelt zu werden. Ohne Wissen über die im eigenen Code-Repository enthaltenen Implikationen ist ein Unternehmen nicht in der Lage, geeignete Maßnahmen zur Reduktion der bestehenden Risiken zu ergreifen.

4.2. Aktive Beiträge in der Open-Source-Community

Weiterhin stellt nicht nur die unkontrollierte und ungeregelte Verwendung von Open-Source-Software ein Problem dar, auch das unkontrollierte und ungeregelte Beitragen zu Open-Source-Projekten birgt verschiedene Geschäftsrisiken. Dabei ist nicht nur die Veröffentlichung von „neuem“ Code unter einer Open-Source-Lizenz zu betrachten, sondern auch Fehlerbehebungen (Bug fixes) sowie „Nicht-Code-Beiträge“ wie etwa ein einem Diskussionsforum.

- Durch Beiträge von Mitarbeitern an Open-Source-Projekten kann dem Unternehmen geistiges Eigentum verloren gehen. Einerseits werden Dritten Nutzungsrechte am Code gewährt, andererseits kann sogar eine Patentlizenz gewährt werden, da durch einige Open-Source-Lizenzen automatisch Patentlizenzen für alle vom Code des Urhebers betroffenen Patente gewährt werden.
- Aus der Art und Struktur der Beiträge, sowohl von funktionalen Erweiterungen als auch von Bugfixes oder Diskussionsbeiträgen, können Dritte Rückschlüsse auf die Produktstrategie des Unternehmens ziehen.
- Durch eine Analyse der Code-Beiträge durch ein Unternehmen an ein Open-Source-Projekt lassen sich unter Umständen Rückschlüsse auf die interne Struktur eines kommerziellen Produkts dieses Unternehmens ziehen.

4.3. Verwicklung in ein Gerichtsverfahren

Eine mögliche Konsequenz ist die Verwicklung eines Softwareunternehmens in ein Gerichtsverfahren, zumeist als Angeklagter.

- Ein Gerichtsverfahren bedeutet erhöhte öffentliche Aufmerksamkeit und in einem Gerichtsverfahren angeklagt zu sein, Open-Source-Software illegal eingesetzt zu haben, bedeutet mit Sicherheit schlechte Presse.
- Die durch ein Gerichtsverfahren verursachen Kosten müssen im Zweifel in voller Höhe vom Unternehmen getragen werden.
- Während eines Verfahrens entsteht eine Arbeitsbelastung (sowohl direkt vor Gericht als Zeuge, als auch im Hintergrund um Unterlagen beizubringen), die zu Lasten der produktiven Zeit geht.

- Durch die Nachweise, die vor Gericht erbracht werden müssen, können Geschäftsinformationen offengelegt werden, die das Unternehmen Wettbewerbern und der allgemeinen Öffentlichkeit nicht zugänglich machen wollte.
- Planungen und Rückstellungen für den möglichen Fall einer Verurteilung verursachen zusätzliche Aufwände.
- Auch wenn eine außergerichtliche Einigung erzielt werden kann, können Aufwände z.B. durch Strafzahlungen, Lizenzgebühren und Überarbeiten oder Neuentwickeln der Software entstehen.
- Schadensersatzzahlungen, Lieferstopp oder Verpflichtung, den Source-Code offenzulegen sind mögliche Auswirkungen einer Verurteilung.

4.4.Verpflichtung, Source-Code offenzulegen

Verliert das Unternehmen das Gerichtsverfahren oder ist es zu einer außergerichtlichen aber für das Unternehmen ungünstigen Einigung gezwungen, so kann eine mögliche Konsequenz sein, bisherigen geschlossenen Produkt-Code offenzulegen.

- Schlechte Presse ist auch unmittelbar zu erwarten, nachdem ein Unternehmen gerichtlich gezwungen wurde, bis dahin geheimen Source-Code offenzulegen. Kunden könnten sich dann fragen, ob das Entdeckte nur die Spitze des Eisbergs ist.
- Der offengelegte Source-Code kann eine nützliche Quelle für Wettbewerber sein, um sich Informationen über Interna der Produkte des verurteilten Unternehmens zu verschaffen. Im schlimmsten Fall müssen wichtige, wettbewerbsdifferenzierende Teile des Codes Dritten zugänglich gemacht werden.
- Auf jeden Fall stellt ein derartiges Urteil Verlust geistigen Eigentums dar. Hier sind neben dem Urheberrecht auch ggf. durch die Open-Source-Lizenz gewährte Patente relevant.
- Wie bei unkontrollierten Beiträgen zu Open-Source-Projekten kann es auch in diesem Fall Wettbewerben möglich sein, Informationen über interne Produktstrukturen zu erlangen.
- Ein Unglück kommt selten allein: Wo eine Rechtsverletzung festgestellt wurde, könnten leicht noch weitere sein. Das kann einzelne Personen, Vereinigungen oder Unternehmen dazu veranlassen, etwas genauer hinzusehen und nach weiteren Lizenzverstößen zu suchen.

4.5.Verurteilung wegen einer Patentverletzung

- Auch Patentverletzungen werden den Ruf des Unternehmens wahrscheinlich nicht positiv beeinflussen.
- Für die Zeit der ungerechtfertigten Nutzung patentierten Codes muss Schadensersatz an den Patentinhaber gezahlt werden. Dieser Schadensersatz kann leicht über die eigentlichen Lizenzgebühren hinausgehen.
- Neben den Zahlungen für die Vergangenheit entstehen Aufwände für die Zukunft, denn entweder müssen Lizenzgebühren für den patentierten Code entrichtet werden oder die entsprechende Komponente muss ausgebaut und eine alternative Lösung implementiert werden.

5. Beispiele für Erfolgsmethoden

Erfolgsmethoden (engl. „best practices“) sind Regeln, Methoden oder Verfahrensvorschriften, welche den Stand der Kunst als bestmögliche Lösung vorliegender Probleme rezipieren. Auf Basis der Interviews sowie der in Kapitel 4 genannten Literatur konnten wir die folgenden Kategorien an Erfolgsmethoden identifizieren:

- Erfolgsmethoden der eigenen Entwicklung: Entwicklerausbildung, Definierte Repositories, Code-Scanning, etc.
- Erfolgsmethoden des Lieferantenmanagements: Vertragsgestaltung, Trust-but-verify, Lieferanten-Audits, etc.

- Erfolgsmethoden zur Erstellung eines Regelwerks (einer Policy): Lizenzklassen, Nutzungsarten, Beiträge zu Projekten, etc.

Alle Erfolgsmethoden hier darzustellen würde den Rahmen des Artikels sprengen. Deswegen haben wir einige exemplarische Erfolgsmethoden ausgewählt. Für weitere Beispiele verweisen wir auf unsere frühere Arbeit [Helmreich 2011].

Verschiedene Erfolgsmethoden ermöglichen Unternehmen, Open-Source-Software gewinnbringend in ihre kommerziellen Produkte zu integrieren. Grundvoraussetzung ist das Verständnis der „Open-Source-Welt", vor allem der rechtlichen Aspekte, die nicht immer intuitiv und direkt verständlich sind. Darauf aufbauend hilft eine Open-Source-Strategie, die Prozesse und Regelungen im Unternehmen konsistent auszurichten. Dabei sind die grundsätzlichen Fragestellungen längst nicht nur technischer Natur, sondern vielmehr weitreichende Geschäftsentscheidungen. Wie positioniert sich ein Unternehmen in Hinblick auf Open-Source-Software? Nachdem Open-Source-Software für ein Unternehmen, das Software entwickelt oder integriert täglich präsent ist, hat die Strategie Auswirkungen auf viele interne Prozesse. Genauso wichtig kann die Außenwirkung sein. Möchte ein Unternehmen, das Open-Source-Komponenten einsetzt und gelegentlich daran Fehler korrigiert, von außen als Softwarehaus gesehen werden, obwohl das Kerngeschäft ein ganz anderes ist? Wie wird mit Open-Source-Software umgegangen? Reagiert das Unternehmen nur auf Ereignisse oder gestaltet es proaktiv?

Die Erfolgsmethoden lassen sich in drei Bereiche einteilen: Policy – Alles rund um Regelungen und Vorschriften. Die Policy bildet das Kernstück, sie stellt die konkrete Ausprägung der Strategie dar. Prozesse stellen die effektive Einhaltung der Policy sicher. Geeignete Infrastruktur unterstützt die Prozesse und steigert deren Effizienz. Nicht immer lassen sich die drei Bereiche scharf abgrenzen.

Exemplarisch stellen wir drei Erfolgsmethoden vor, die entscheidende Auswirkungen auf die erfolgreiche und kontrollierte Integration von Open-Source-Komponenten in kommerzielle Software haben.:

5.1.Überwachung der Lieferantenschnittstelle

Da viele Unternehmen Software nicht isoliert entwickeln, sondern (teilweise große) Teile von anderen Firmen zugeliefert werden, sind Regelungen an der Lieferantenschnittstelle sehr wichtig. Dabei ist an der Lieferantenschnittstelle mehr als der naheliegende Fall „Lieferant übergibt Software" zu betrachten.

Von Bedeutung ist bereits die Übergabe von bereits vorhandenem Code oder Schnittstellendefinitionen an den Lieferanten, der an diesen weiterarbeiten soll. Hier muss geprüft und entschieden werden, was unter welchen Bedingungen an den Lieferanten weitergegeben wird.

Im Vertrag sollte vereinbart sein, was der Lieferant während seiner Arbeit in Bezug auf Open-Source-Software zu berücksichtigen hat. In welchem Umfang ist die Integration von Open-Source-Komponenten gestattet oder gewünscht? Wie sind Entscheidungen über einen solchen Einsatz zu treffen? In welchen Fällen muss der Auftraggeber involviert werden? Unter welchen Bedingungen dürfen Mitarbeiter des Zulieferers Beiträge an die Open-Source-Community zurückgeben? Unter Umständen kann es sinnvoll und notwendig sein, dem Lieferanten vorzuschreiben, bestimmte vom Auftraggeber vorgegebene (Teil-)Prozesse einzuhalten.

Der entscheidende Punkt ist natürlich die Übergabe der Lieferobjekte an den Auftraggeber. Welche Dokumente werden außer dem Source-Code übergeben? Werden nötige Open-Source-Verwendungsnachweise vom Zulieferer oder von Auftraggeber erstellt? Gibt es Open-Source-Komponenten, bei deren Verwendung der Auftraggeber die Abnahme verweigert? Wird der gelieferte Code überprüft?

Auch für die Zeit nach der Auslieferung sind vertragliche Regelungen sinnvoll, vor allem hinsichtlich der Haftung und Gewährleistung. Wer ist verantwortlich, wenn technische Probleme mit Open-Source-Komponenten auftreten (deren Lizenzbedingungen schließen in der Regel jegliche Haftung aus)? Wer ist verantwortlich, wenn lizenzrechtliche Probleme auftreten?

5.2.Einsatz von Code-Scannern

Die oben aufgeworfene Frage „Wird der gelieferte Code überprüft?" schneidet eine Thematik an, die nicht nur an der Lieferantenschnittstelle relevant ist. Auch wenn alle Beteiligten ihr Möglichstes geben, Open-Source-Software nur lizenz- und regelungskonform zu verwenden und jede Verwendung ordentlich zu dokumentieren,

kann in der Regel niemand garantieren, dass die erstellte Dokumentation über die Verwendung von Open-Source-Komponenten auch nur ansatzweise vollständig ist. Dabei ist ein derart disziplinierter Umgang mit Open-Source-Software eher die Ausnahme. Open-Source-Code kann außer durch explizite genehmigte und deklarierte Verwendung auch auf viele andere Art und Weisen in die Code-Basis eines Unternehmens gelangen.

Eine Überprüfung von Hand ist nur bei sehr geringem Code-Volumen durchführbar. Diese Überprüfung kann mit entsprechenden Tools automatisiert werden. Diese Werkzeuge sind in der Lage sowohl in Source Code als auch in Binärdateien Open-Source-Anteile aufzudecken, auch wenn nicht vollständige Bibliotheken oder Dateien gefunden wurden, also zum Beispiel wenn ein Entwickler ein urheberrechtlich geschütztes Code-Stück aus einem Buch abgeschrieben hat (dessen Verwendung dort vermutlich lizenzrechtlich unproblematisch ist). Durch den Einsatz von derartigen Scannern kann ein Unternehmen eine wesentlich höhere Gewissheit (wenn auch keine absolute) über die in seinem Code-Repository enthaltenen Open-Source-Komponenten erhalten. Teilweise lassen sich die Tools auch in bestehende Werkzeugketten einbinden, so dass ein Scan im bestehenden Prozess automatisch angestoßen werden kann.

5.3. Entwicklerausbildung

Wie vieles im Software-Umfeld hängt die erfolgreiche Implementierung und Umsetzung von Regelungen und Prozessen für Open-Source im Wesentlichen von der Ausbildung der Mitarbeiter ab. Alle Beteiligten müssen ein grundlegendes Verständnis für die Zusammenhänge und Risiken beim Einsatz von Open-Source Komponenten kennen. Dabei ist eine einmalige Informationsveranstaltung für Entwickler zwar sehr wichtig, aber in zwei Dimensionen zu kurz gedacht: Einerseits müssen nicht nur Entwickler über die Implikationen von Open-Source Bescheid wissen, andererseits muss sichergestellt sein, dass jeder neue Mitarbeiter auch geschult wird und der Wissensstand aller Mitarbeiter aktuell ist. So sollte bei Schulungen auch das Management nicht außen vor bleiben und die Rechtabteilung genauso involviert sein. Im gesamten Unternehmen sollte Open-Source mit allen Vorteilen und Risiken bewusst wahrgenommen werden und die Risiken als potentiell geschäftsschädigend erkannt werden.

6. Integration in den Entwicklungszyklus

Open-Source ist aus der Softwareentwicklung nicht mehr wegzudenken. Für viele Anwendungsgebiete gibt es etablierte Open-Source-Komponenten, die jeder kostenfrei aus dem Internet herunterladen kann. Entwickler tun dies. Es stellt sich schon lange nicht mehr die Frage, ob Open-Source eingesetzt wird, sondern wie. Die Risiken, die die Nutzung des großen Potentials von Open-Source-Software birgt, sind in ihrer Vielfalt und ihrem Ausmaß überschaubar. Bei entsprechend konsequentem kontrollierten und geregelten Einsatz lässt sich (fast) jedes Risiko, das durch Einsatz von Open-Source-komponenten entstanden ist, sowohl genau beschreiben als auch hinsichtlich der Kosten der Risikovermeidung (im Zweifel Ersetzen der Komponente durch eine Eigenentwicklung oder ein kommerzielles Drittprodukt) sehr genau beziffern. Nach einem initialen Aufwand (Inventarisierung der bestehenden Code-Basis) lassen sich die notwendigen Aktivitäten gut in bestehende Prozesse im Entwicklungszyklus integrieren.

7. Literatur

[Badenhope 2010] Badenhope, V.. Pulling It All Together: An Open Source Action Plan for In–House Counsel. Open Source Business Conference, 18.03.2010.

[Black Duck Software 2010] Black Duck Software: Diverse Whitepaper, http://www.blackducksoftware.com/resources/whitepapers, Zugriff am 1.12.2010.

[Hammond 2009] Hammond, J.. Best Practices: Improve Development Effectiveness Through Strategic Adoption Of Open Source, Forrester, 2009.

[Helmreich 2011] Helmreich, M.: Best Practices of Adopting Open Source Software in Closed Source Products, Nürnberg, 2011, http://osr.cs.fau.de/2011/05/23/osr-group-thesis-best-practices-of-adopting-open-source-software-in-closed-source-software-products-in-english/; Zugriff am 1.4.2011.

[OpenLogic 2010a] OpenLogic: Diverse Webinare, http://www.openlogic.com/downloads/webinars.php; Zugriff am 1.12.2010.

[OpenLogic 2010b] OpenLogic: Diverse Whitepaper, http://www.openlogic.com/downloads/whitepapers.php; Zugriff am 1.12.2010.

[OpenBRR 2005] OpenBRR.org: Business Readiness Rating for Open Source, http://www.openbrr.org Zugriff 2010 über archive.org

[Radcliffe 2010] Radcliffe, M.: Managing Open Source Software 2010: Best Practices, Open Source Business Conference, 17.03.2010.

[Sutor 2010] Sutor, B.: Asking the Hard Questions about Open Source Software. Open Source Business Conference, 17.03.2010.

[Strauss et al. 1996] Strauss, A.; Corbin, J.; Niewiarra, S.: Legewie, H.: Grounded theory: Grundlagen qualitativer Sozialforschung. Beltz Psychologie Verl.-Union., Weinheim, 1996.