Einsatz von Open Source Software in öffentlichen Behörden und Unternehmen:

Lohnt ein Umstieg von proprietärer Software hin zur Open Source Software?

Norman Schwaneberg
Studiengang: Wirtschaftsinformatik

Hochschule Wismar, Fachbereich Wirtschaft
Philipp-Müller-Straße 21, D-23952 Wismar, Deutschland

ABSTRACT: Öffentliche Behörden, Manager von Unternehmen, Administratoren, Programmentwickler und Programmanwender werden immer häufiger mit Open-Source-Software (OSS) konfrontiert. Der Marktanteil von OSS ist in den letzten Jahren kontinuierlich gestiegen. Daher ist es wichtig, die wesentlichen Unterschiede zwischen proprietärer Software und Open Source Software genau zu kennen. Der Autor führt Sie an das Thema durch die Entstehungsgeschichte heran und bespricht die wesentlichen Einflussgrößen, wie zum Beispiel die verschiedenen Softwarelizenzen. Des Weiteren werden die Chancen und Risiken von OSS erläutert. Es wird eine Hilfestellung für eine Entscheidung zwischen OSS und proprietärer Software gegeben. Der Beitrag gibt den Forschungsstand hinsichtlich der Wirtschaftlichkeit von Open Source Software wider, gibt einen Ausblick auf die zukünftige Entwicklung und richtet sich an Praktiker und Interessierte, die überlegen, auf Open Source Software umzusteigen.

Software users and developers have to deal with Open Source Software (OSS) more and more. The market share of OSS increased significantly over the past years. The author introduces the subject by looking at the genesis of OSS and the main determining factors for its development. Furthermore, the risks and chances of OSS will be discussed. The article reflects the state of research regarding the efficiency of OSS and is directed to practitioners and other interested parties who consider migrating to OSS. They will receive advice for selecting between OSS and proprietary software.

1 ENTSTEHUNGSGESCHICHTE

Die Entstehungsgeschichte der Open Source Software (OSS) reicht bis in die frühen Anfänge der Computerindustrie, in den Jahren 1960 bis 1970, zurück. Zu dieser Zeit galt Software als frei und nur als Nebenprodukt, welches als Obolus zur Hardware dazu gegeben wurde. Es wurde als selbstverständlich angesehen, dass jeder Zweite oder Dritte einen bereits implementierten Algorithmus nicht neu entwickeln muss. Die Entwickler haben den betreffenden Quellcode ohne Auflagen weitergegeben. Unfrei wurde Software erst durch Microsoftgründer Bill Gates und AT&T (*American Telephone & Telegraph Corporation*) in Amerika. Die beiden Unternehmen haben als erstes erkannt, dass sich Software kostengünstig erstellen und gewinnbringend vermarkten lässt. 1983 veröffentlichte AT&T das UNIX System V als proprietäre Software und stellte die Nutzung unter Lizenzrecht. Im selben Jahr widersetzten sich andere Softwareentwickler, wie zum Beispiel Richard Stallmann und beruhten sich auf das Prinzip der Sechziger Jahre: "When we speak of free software, we are referring to freedom, not price" und in den achtziger Jahren veredelte er seine Aussage zu „Don't think free as in free beer; think free as in free speech“. Dadurch entstand der Begriff „Freie Software“. Ferner begann Stallmann 1983 das GNU-Projekt, welches sich als Ziel setzte, ein freies Betriebssystem zu entwickeln. GNU ist ein rekursives Akronym („GNU’s not Unix“), dass dadurch auf die Ähnlichkeit mit UNIX, als auch auf die Abgrenzung zu unfreien UNIX-Systemen, anspielt. Im Jahre 1985 wurde die Free Software Foundation (FSF) gegründet, die später über die GPL (General Public License) wacht. Stallmann und Jerry Cohen veröffentlichen 1989 die erste Version der GPL. 1998 entstand eine heftige Diskussion bei vielen Entwicklern, ob der Begriff der Freien Software (Free Software) genau das definiert, welches ursprünglich gemeint ist. In der englischen Sprache bedeutet der Begriff „free“: frei und kostenlos. Jedoch ist OSS nicht kostenlos, da zum Beispiel beim freien Download Kosten entstehen. Die Entwickler leisten größtenteils freiwillige arbeit oder es arbeiten Entwickler aus Projekten von Unternehmen an der Software, die trotzdem möglicherweise ein kommerzielles Ziel beabsichtigen. Daher konnte man sich nicht mit dem Begriff „Free Software“ anfreunden. Um Freie Software mit Marketing technischen mitteln geschickt vermarkten zu können und dem ideologischen Streit zwischen den Entwicklern entgegenzuwirken, wurde von Christine Peterson vom Foresight Institute der Begriff „Open Source“ vorgeschlagen. Eric Steven Raymond, Bruce Perens und Tim O’Reilly bestätigten den Vorschlag und nutzten den Begriff für die Open Source Initiative (OSI). Allerdings lehnt der Gründer der FSF Richard Stallmann die Formulierung strikt ab. Um der anscheinend niemals endenden Diskussion zu beenden, wurde zuerst das Akronym FOSS und dann später FLOSS eingeführt. FOSS steht für „Free Open Source Software“ und FLOSS „Free / libre Open Source Software”. Das L wurde beigefügt, damit auch in anderen Ländern wie Frankreich (libre), Spanien (libre), Portugal (livre) oder Italien (libero) unmissverständlich das gleiche, also „frei“ und nicht kostenlos, verstanden wird. Somit wird seitdem endgültig der Streit zwischen der Free Software Foundation und der Open Source Initiative aus dem Weg gegangen und die Anhänger beider Parteien zufrieden zu stellen. Hauptsächlich wurde FLOSS erschaffen, damit vorhandene Quellcodes nicht wegen Geheimhaltung wie es bei proprietärer Software der Fall ist, erneut programmiert werden muss, und ergänzend hoch qualitative Software sicher gestellt werden kann.

2 BEGRIFFSDEFINITIONEN

2.1 Proprietäre Software

Bei proprietärer Software hält der Entwickler, das Entwicklerteam oder ein Unternehmen sich das exklusive Recht an der Software vor und verbietet das Kopieren, Weitergeben, Verändern oder Studieren des Quellcodes. Weiterhin wird der Quelltext geheim gehalten und nur unter bestimmten eingegrenzten Bedingungen verfügbar gemacht (Beispiel: durch richterlichen Beschluss bei Microsoft). Daher wird die Softwareart als „Closed Source“ bezeichnet. Proprietäre Software besitzt oftmals keine offenen Standards, dass heißt andere Software kann das Dateiformat, das Protokoll oder eine bestimmte Schnittstelle nicht ohne weiteres einfach benutzen. Es wird die Interoperabilität zu anderen Softwareherstellern vermieden, damit der Hersteller seine Monopolstellung erreichen oder ausbauen kann und die Anwender von dem Unternehmen abhängig werden. Unter Interoperabilität verstehen Informatiker das Zusammenspiel zwischen verschiedener Software, die das gleiche Dateiformat oder Protokoll verwenden. Außerdem ist diese Softwareart an bestimmte Nutzungsrechte gebunden, welche gegen eine Bezahlung einer Lizenzgebühr eingeschränkt nutzbar wird.

2.2 Free / libre Open Source Software

Wie der Begriff Free / libre Open Source Software (FLOSS) entstanden ist, wurde bereits in der Entstehungsgeschichte erläutert. Was das Prinzip genau definiert wird anhand der folgenden Kriterien dargestellt:

- Freier Zugang zum Quellcode
- Freie Weitergabe der Software
- Möglichkeit der beliebigen Veränderung der Software
- Keine eingeschränkte Nutzung

Unter freiem Zugang zum Quellcode wird die vollständige Offenlegung des gesamten Quellcodes der Software verstanden. Bei einer ausführbaren Binärdatei muss der Quellcode beigefügt werden. Eine Binärdatei ist eine in Maschinencode übersetzte Quelldatei, die nachträglich vom Menschen nicht mehr lesbar ist. Wenn es aus irgendeinem Grund nicht realisierbar ist, dass die Quelldatei nicht beigefügt werden kann, dann muss der Quellcode zum Beispiel durch Veröffentlichung auf der entsprechenden Internetseite öffentlich zugänglich gemacht werden. Bei dem Kriterium der freien Weitergabe der Software muss die FLOSS an beliebige Dritte weitergegeben werden dürfen und eine Einschränkung durch die Entwickler der Applikation darf nicht erfolgen. Lizenzgebühren zu erheben ist dem Autor des Quellcodes nicht ausdrücklich verboten, jedoch durch die freie Weitergabe der Software als surreal anzusehen. Bei dem dritten Merkmal muss der Programmierer der FLOSS eine beliebige Modifikation an dem Quellcode erlauben, allerdings muss das veränderte Programm unter den gleichen Lizenzbedingungen wie das Original angeboten werden. Der Urheber der originalen Software kann jedoch verlangen, dass die veränderte Applikation als separater Patch angeboten werden muss. Unter dem Begriff Patch wird ein kleines Softwareupdate oder eine Softwarekorrektur verstanden. Keine eingeschränkte Nutzung bedeutet, dass die FLOSS für beliebige Zwecke von jedem genutzt werden kann. Des Weiteren ist eine Einschränkung der kommerziellen Nutzung oder der Nutzung durch bestimmte Personen nicht erlaubt. Außerdem darf die Applikation weder an den Einsatz einer anderen Software, noch an den nicht Einsatz einer anderen Software gebunden sein. Auch eine Bedingung der Geheimhaltung ist nicht zulässig. Die soeben genannte Bedingung wurde aufgeführt, da Microsoft eine Alternative durch „Shared Source“ zu Open Source versucht ins Leben zu rufen, welches mit dem Open Source Gedanken nicht vereinbar ist (Quelle: Fraunhofer Institut, S.13).

3 LIZENZMODELLE VON FLOSS

Im Wesentlichen können FLOSS Lizenzmodelle in drei Kategorien eingeordnet werden.

- starkes Copyleft
- schwaches Copyleft
- kein Copyleft

Richard Stallmann prägte den Begriff „Copyleft“ in Anlehnung an das englische Wort Copyright. Das Copyright erlaubt es nur dem Urheber die Daten zu verbreiten und dient dazu, die geistigen Erzeugnisse zu schützen. Beim Copyleft Verfahren ist eine uneingeschränkte Verbreitung von Kopien und modifizierten Versionen eines Softwareerzeugnisses möglich. Die einzige Bedingung die gestellt wird ist, dass man generell jede Modifikation und Weiterentwicklung von FLOSS nur unter derselben Lizenz, als freie Software, weiter geben darf. Die bekannteste Copyleft-Lizenz ist die GNU General Public License (GPL).

Art des Copyleft	starkes CL	schwaches CL	Kein CL
Kombinations-möglichkeit mit proprietärer Software	Keine Einbindung in prop. Code möglich	Statisches und dynamisches Linken von Code mit prop. SW möglich Eigen-entwicklungen dürfen als proprietäre SW verbreitet werden	Keine Vorgaben Der gesamte Code darf auch als proprietäre Software weitergegeben werden
Beispiel-Lizenz	GPL	LGPL, MPL	BSD, Apache

Tabelle 1: Eigenschaften von Copyleft (http://www.heise.de/open/artikel/75786/0)

3.1 Starkes Copyleft

Bei starkem Copyleft gelten für alle Modifikationen und Weiterentwicklungen des Programms dieselben Lizenzbedingungen wie für den originalen Quellcode. Damit soll sichergestellt werden, dass der einmal als FLOSS freigegebener Quellcode frei bleibt und nicht als ein proprietäres Produkt verwendet werden kann. Die bekannteste Copyleftvariante ist die GNU General Public License (GPL). Sie gestattet dem Anwender die Verwendung, Modifikation und Weitergabe des Quellcodes, solange dem Empfänger dieselben Rechte gewährt werden. Änderungen gegenüber dem originalen Quellcode müssen ausdrücklich gekennzeichnet werden. Eine Weitergabe des modifizierten Quellcodes setzt voraus, dass das Programm nicht nur als Binärdatei zur Verfügung gestellt wird, sondern auch der Quellcode beiliegt oder auf anderem Wege veröffentlicht wird. Die GPL schließt einen Haftungsausschluss ein, die dem

Distributor für das Produkt die Möglichkeit einräumt, gegen eine Bezahlung, eine Garantie dem Anwender gewährt.

3.2 Schwaches Copyleft

Durch die FSF wurde mit der GNU Lesser General Public License (LGPL) eine schwache Copyleftform geschaffen.
Das L in LGPL stand anfänglich für Library (Bibliothek). Der Name wurde jedoch von der FSF in Lesser (weniger) umbenannt, um den Entwickler indirekt einen Hinweis zu geben, dass die Entwickler besser GPL anstatt LGPL nutzen sollen. Die LGPL ermöglicht die Kombination von proprietärem Quellcode und freiem Binärcode (jedoch nicht mit freiem Quellcode). Wenn ein Programm eine Systembibliothek benutzt, die unter der GPL Lizenz steht, dann muss das Programm ebenfalls unter der GPL gestellt werden. Eigenentwicklungen können als proprietäre Software vermarktet werden. Ein Beispiel für ein schwaches Copyleft ist die Mozilla Public License (MPL). Bei dieser Lizenz dürfen unabhängige Erweiterungen (add-on) und Neuentwicklungen (redevelopment) auch unter einer anderen Lizenz wie zum Beispiel der proprietären Lizenz vermarktet werden.

3.3 Kein Copyleft

Eine Non-Copyleft-Lizenz (kein Copyleft) beinhaltet keine Vorschrift, wie Modifikationen und Weiterentwicklungen weitergegeben werden müssen. Daher ist es jedem Entwickler erlaubt Quellcode beliebig einzusetzen, zu modifizieren und weiter zu entwickeln. Sie fordern jedoch einen Copyrighthinweis und einen Haftungsausschluss. Non-Copyleft-Lizenzen sind stark an die Berkley Software Distribution (BSD) angelehnt.

4 CHANCEN VON FLOSS

4.1 Anpassbarkeit

Jeder Anwender kann für seine individuellen Zwecke die FLOSS anpassen oder erweitern. Die benötigten Programme brauchen daher nicht vollständig neu entwickelt werden. Der eigene Quellcode kann somit ohne Probleme das vorhandene FLOSS Grundprodukt ergänzen.

4.2 Quellcode ist wieder verwendbar

Durch die Verwendung von vorhandener Quellcodes können Entwicklungskosten und Entwicklungsphasen eingespart werden. Durch das Studium des bereits existierenden Quellcodes findet eine Übertragung des Know-hows statt.

4.3 Höhere Produktqualität

Die Befürworter von FLOSS behaupten, dass diese Softwareart tendenziell eine höhere Produktqualität besitzt, als proprietäre Software. Diese These wird dadurch begründet, dass FLOSS keinerlei Marktzwängen unterliegt, wie es zum Beispiel bei kommerzieller Software durch feste Veröffentlichungstermine der Fall ist.

4.4 Anbieterunabhängigkeit

Dadurch das FLOSS offene Standards verwendet, wird kein Anwender in eine gewisse Abhängigkeit zu einem bestimmten Hersteller gezwungen.

4.5 Höhere Sicherheit

Da jeder Anwender die Möglichkeit hat den Quellcode zu studieren, können Sicherheitslücken schneller erkannt werden. Daher wird FLOSS Produkten ein höheres Maß an Sicherheit zugesprochen.

4.6 Offene Standards

Die verwendeten Dateiformate, Dateiaustauschstandards und Protokolle sind als offene Standards festgelegt. Deshalb wird die Interoperabilität zwischen FLOSS Produkten gewährleistet.

4.7 Keine Lizenzkosten

Es fallen bei FLOSS Produkten keine Lizenzkosten an.
Die Gesamtheit der Lizenzkosten, die über den kompletten Lebenszyklus wegfallen, stellen einen geringen Anteil dar.

5. RISIKEN VON FLOSS

5.1 Gewährleistungsrechte

Die Produktanwender können in der Regel keine Gewährleistungs- und / oder Haftungsansprüche gegenüber Entwickler stellen. Weiterhin steht in den meisten Lizenzen, dass keine Garantie für die Funktionstüchtigkeit gegeben wird. Der Anwender allein trägt das volle Risiko.

5.2 Kein Support durch Entwickler

Die Programmentwickler gegeben in den meisten Fällen keinen Support für die erstellte Software. Es bleibt nur die Möglichkeit Dienstleistungen Dritter Anbieter in Anspruch zu nehmen.

5.3 Höherer Schulungsaufwand

Mitarbeiter von Unternehmen oder Behörden müssen oftmals umgeschult werden. Eventuell muss qualifiziertes Fachpersonal eingestellt werden. Während der Umstellungsphase sinkt die Arbeitsproduktivität.

5.4 Ungewisse Weiterentwicklung

FLOSS Projekte müssen nicht weiter entwickelt werden und können jederzeit aufgegeben werden. Außerdem sind die Entwickler nicht verpflichtet die Software zu warten oder zu pflegen.

5.5 Benötigte Applikationen sind nicht verfügbar

Zurzeit sind weniger Applikationen verfügbar als für Windowssysteme, dass ist oftmals ein Grund warum nicht FLOSS nicht eingesetzt wird.

5.6 Mangelnde Interoperabilität mit proprietärer SW

Kommerzielle Softwareanbieter haben meistens kein Interesse an einer Interoperabilität zu anderen Produkten, um den Umstieg auf einen anderen Anbieter zu erschweren. Ein bekanntes Beispiel ist Microsoft Office und Open Office.

Die Tabelle 2 führt die Chancen und Risiken von FLOSS übersichtlich auf:

Chancen	Risiken
Anpassbarkeit	Keine Gewährleistungsgarantie
Quellcode ist wieder verwendbar	(oft) kein Support durch Entwickler
Höhere Produktqualität	Höherer Schulungsaufwand
Anbieterunabhängigkeit	Ungewisse Weiterentwicklung
Höhere Sicherheit	Applikationen teilweise nicht erhältlich
Offene Standards	Teilweise mangelnde Interoperabilität mit kommerzieller Software
Keine Lizenzkosten	

Tabelle 2: Chancen und Risiken von FLOSS
(http://www.e-business.iao.fraunhofer.de/docs/FhG_OSS_Studie.pdf)

6 WIRTSCHAFTLICHKEIT VON FLOSS

Die Wirtschaftlichkeit von FLOSS gegenüber proprietärer Software kann nicht allgemeingültig bewiesen werden, da sie immer vom Einzelfall abhängig ist. Daher empfiehlt das Fraunhofer Institut immer eine Wirtschaftlichkeitsprüfung vor der Einführung bzw. dem Umstieg durchzuführen.
Das Fraunhofer IAO sechs Phasen Modell zur Wirtschaftlichkeitsprüfung und Strategieoptimierung durchläuft:

1. IST-Analyse
2. Strategiebewertung
3. Identifikation zusätzlicher Anwendungen
4. Identifikation von Handlungsalternativen
5. Bewertung und Abgabe einer Handlungsempfehlung
6. Konzeption und Umsetzung

Die Wirtschaftlichkeitsprüfung stellt die in Frage kommende FLOSS der proprietären Software gegenüber und bewertet beide kritisch. Dabei werden nicht nur die Einsparmöglichkeiten durch die Unterschiede in den Lizenzen geprüft, sondern auch die quantitativ schwer zu erfassenden Kriterien, wie zum Beispiel die Freiheitsrechte, die für Unternehmen und öffentliche Behörden einen strategischen Mehrwert bilden. Das folgende Kriterienraster dient zur Bewertung von Softwareprodukten:

Einführungskosten	Betriebskosten	Strat. Kriterien
Personalkosten	Personalkosten	Stabilität
Beratungskosten	Wartung	Sicherheit
Linzenzkosten	Hardwarekosten	Hersteller-unabhängigkeit
Schulungskosten	laufende Schulungskosten	Benutzerfreundlichkeit
Migrationskosten	Updatekosten	Interoperabilität
Installationskosten		Anpassbarkeit an individuelle Zwecke
Einarbeitungsaufwand		vorhandenes Know-how

Tabelle 3: Kriterienraster zur Bewertung von SW-Produkten
(http://www.e-business.iao.fraunhofer.de/docs/FhG_OSS_Studie.pdf)

Die EU-Studie der UNU-MERIT mit dem Titel "Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU" hat veröffentlicht, dass Einsparmöglichkeiten bis zu 36% möglich sind. Den wirtschaftlichsten Vorteil haben dabei größere Unternehmen, da meistens mehrere lizenzpflichtige Arbeitsplätze vorhanden sind, als bei kleineren Unternehmen. Ein besonderes Augenmerk auf FLOSS wurde bei Unternehmen und öffentlichen Behörden in den Bereichen Büroanwendungen, Server-Betriebssysteme, Datenbanken und Content Management Systemen geweckt, da die Entwicklung der Software sehr vorangeschritten und bereits qualitativ hochwertig ist.

7 SCHLUSSFOLGERUNGEN UND AUSBLICK

Bei der Auswahl von Softwareprodukten müssen immer die beiden Softwarearten mittels Wirtschaftlichkeitsprüfung und Nutzwertanalyse gegenüber gestellt werden. Für den jeweiligen Einzelfall kann es zu einer Entscheidung für die FLOSS führen. Bei Unternehmen oder Behörden kann es zu klaren wirtschaftlichen, sowie strategischen Vorteilen führen. Zum Beispiel: Die Münchener Stadtverwaltung hat mit ihrem Linux-Projekt „LiMux" erfolgreich bewiesen, dass eine Umstellung von Windows auf Open Source realisierbar ist. Der Marktanteil von FLOSS Produkten wird in den nächsten Jahren wahrscheinlich noch weiter ausgebaut werden, besonders bei Serverprodukten wie Datenbanken oder Server-Betriebssystemen. Desktop Produkte werden für die Entwicklung vermutlich noch etwas mehr Zeit benötigen, da in diesem Bereich proprietäre Software noch sehr stark verbreitet ist. Weiterhin haben Anbieter von proprietärer Software in der Vergangenheit offene Standards vermieden. Jedoch scheint Microsoft erkannt zu haben, dass Interoperabilität zwischen MS Office und Open Office immer mehr an Bedeutung gewinnt und so hat Microsoft einen Abkommen mit Novell getroffen, um an diesem Problem zu arbeiten. Novell will den Quellcode schreiben, um Office Open XML in Open Office zu unterstützen und dann ein Plugin für ihre Open Office Version bereitstellen. Des Weiteren wird es von Novell Software geben, um virtualisierte Windows Systeme zu verwalten. Microsoft dagegen wird Lösungen anbieten, um virtuelle Linux Systeme zu managen.

8 REFERENZEN

1. Gosh, R. A.: EU-Studie, "Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU", http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf, letzter Zugriff 20.02.2007
2. Günther, A.: Open-Source-Software und die GPL, in: T3N Magazin, 04.2006, S. 93-95
3. Kett, H.; Renner, T; Rex, S.; Vetter, M.: Eine Studie der Faunhofer-Gesellschaft, OSS, Einsatzpotenziale und Wirtschaftlichkeit, http://www.e-business.iao.fraunhofer.de/docs/FhG_OSS_Studie.pdf, letzter Zugriff: 20.02.2007
4. o. V.: Open-Source-News: Microsoft schließt Pakt mit Novell, in: T3N Magazin, 04.2006, S. 6
5. Williams, S..: „Free as in Freedom", O'Reilly USA 2002 http://www.oreilly.com/openbook/freedom/ch09.html