Open Source Software-Release:
Ein Ratgeber für die Veröffentlichung von Software unter dem Open Source-Status

Heidi Hohensohn, Ulrich Bretschneider, Stefan Renk

C-LAB Report

Vol. 3 (2004) No. 1

Cooperative Computing & Communication Laboratory

ISSN 1619-7879

C-LAB ist eine Kooperation
der Universität Paderborn und der Siemens Business Services GmbH & Co OHG
www.c-lab.de
info@c-lab.de

C-LAB Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens Business Services GmbH & Co OHG

Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens Business Services GmbH & Co OHG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens Business Services GmbH & Co OHG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1619-7879

C-LAB
Fürstenallee 11
33102 Paderborn
fon: +49 5251 60 60 60
fax: +49 5251 60 60 66
email: info@c-lab.de
Internet: www.c-lab.de

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
AGB	Allgemeine Geschäftsbedingungen
BGB	Bürgerliches Gesetzbuch
CVS	Concurrent Versioning System
d.h.	das heißt
etc.	et cetera
f.	folgend
ff.	fortfolgend
FSF	Free Software Foundation
Nr.	Nummer
o.ä.	oder ähnlich
o. J.	ohne Jahrgang
OS	Open Source
OSD	Open Source Definition
OSI	Open Source Initiative
OSS	Open Source Software
o.V.	ohne Verfasser
s.	siehe
S.	Seite
SW	Software
Tab.	Tabelle
UrhG	Urhebergesetz
u.a.	unter anderem
vgl.	vergleiche

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Softwareindustrie stand dem Open Source (OS) -Gedanken vielfach grundsätzlich skeptisch gegenüber, steuert er doch anscheinend gegen die kommerziellen Interessen dieses Industriezweiges. Wie es scheint, besitzt die Open Source-Idee aber das Potenzial, den Skeptikern den Wind aus den Segeln zu nehmen. Doch während sich noch einige Giganten der Softwareindustrie gegen Open Source Software (OSS) vehement zur Wehr setzen und sich in ihrer Existenz bedroht fühlen, erkennen andere die Potenziale und stehen dieser Idee nicht mehr grundsätzlich kritisch gegenüber. Im Gegenteil: Open Source wird von diesen Vertretern eher als eine strategische Option begriffen. Und so tun diese Softwareunternehmen etwas, was auf den ersten Blick für Marktwirtschaftler völlig abwegig erscheinen mag. Sie veröffentlichen ihre in der Regel vormals proprietäre, für Geld angebotene Software unter dem Status Open Source. Open Source Software wird in Unternehmenskreisen also immer öfter als Strategie beispielsweise zur Schaffung von Lock-in-Effekten und Kaufanreizen sowie/oder zur Steigerung der eigenen Reputation verstanden.

Aber auch für nicht-kommerzielle Softwareentwickler, wie zum Beispiel Universitäten, bietet der Open Source-Gedanke Perspektiven. So wird beispielsweise Software, die im Rahmen von Forschungsarbeiten entstand, als Open Source Software veröffentlicht. Auf diese Weise verstauben die Forschungsergebnisse nicht in den Schubladen und können der Öffentlichkeit zugänglich gemacht werden. Auch entfällt ein aufwendiges und kostenintensives Vertriebsmanagement im Falle einer Vermarktung dieser Forschungsprojekte durch die Universität selbst, für das diese in der Regel keine Mittel zur Verfügung hat. Einen derartigen Aufwand sollten diese Einrichtungen wegen ihrer Stellung in der Öffentlichkeit auch ohnehin nicht leisten dürfen.

An dieses Publikum der kommerziellen und nicht-kommerziellen Softwareentwickler richtet sich die vorliegende Arbeit. Sie soll als Leitfaden für öffentliche Einrichtungen und Softwareunternehmen aus der freien Wirtschaft gleichermaßen dienen, die Software aus nicht-kommerziellen oder kommerziellen Gründen unter dem Status einer OSS veröffentlichen möchten. Dabei tauchen in der Praxis immer wieder anscheinend unüberwindbare Hürden auf. Vor allem die mit einer OS-

Veröffentlichung verbundenen rechtlichen Aspekte stellen Hindernisse dar. Aber auch organisatorischen Fragen, wie der Aufbau einer Entwickler-Community, Lizenzaspekte sowie formelle Anforderungen an eine OSS, stellen potenzielle OSS-Anbieter oft vor unüberbrückbare Herausforderungen.

Die vorliegende Arbeit soll deshalb für diese Zielgruppe einen praktischen Ratgeber auf dem Weg in die OS-Veröffentlichung liefern. Dabei sind der hier vorgeschlagene Ansatz sowie die Arbeit selbst in der Form eines Ablaufdiagramms aufgebaut. Chronologisch geordnet werden die relevanten Schritte eines OS-Veröffentlichungsprozesses in einem jeweils eigenen Kapitel behandelt. Inhaltlich bemüht sich die Arbeit dabei, für jeden Einzelschritt die jeweils gängigsten Lösungskonzepte und aktuell am häufigsten diskutierten Modelle einzubeziehen und teilweise zu übernehmen. Es geht in dieser Arbeit nicht darum, eine breite Darstellung diskutierter Lösungskonzepte aufzuzeigen. Vielmehr soll ein pragmatischer Leitfaden ohne Anspruch auf Vollständigkeit geboten werden. Ziel der Arbeit ist es, einen einfach zu handhabenden Leitfaden für den Prozess der OS-Veröffentlichung von Software bereitzustellen.

Jedes Kapitel ist dabei so aufgebaut, dass zu Beginn eines jeden Schrittes in Kurzform die Ziele, Leitsätze und die Aufgabenträger eines jeden Prozessschrittes dargestellt werden. In einem weiteren Punkt innerhalb jedes Kapitels werden dann ausführliche Erläuterungen gegeben. Ein Ablaufplan im Anhang dieser Arbeit gibt die logischen Abhängigkeiten der einzelnen Prozessschritte grafisch wider. Zudem findet sich im Anhang eine Checkliste, anhand derer der Leser im Falle einer eigenen OS-Veröffentlichung den Prozessablauf praktisch erfassen und kontrollieren kann.

Die Schritte der OS-Veröffentlichung berühren vielfältige Kompetenzen. Dabei sind insbesondere technische, rechtliche und betriebswirtschaftliche Schwerpunkte zu setzen. In der vorliegenden Arbeit wird deshalb unter dem Punkt „Prozessbeteiligte“ eines jeden Schrittes versucht, den Kompetenzbedarf für jeden Prozessschritt abzuschätzen. Für diese Aufwandsschätzungen wurden dabei drei Kategorien unterschieden:

- technische Kompetenz,
- rechtliche Kompetenz und

- betriebswirtschaftliche Kompetenz.

Für jeden einzelnen Prozessschritt wurden dabei insgesamt fünf Punkte auf diese drei Fachkompetenzen verteilt. Auf diese Weise ergab sich je nach Punkteverteilung für jeden Prozessschritt eine mehr oder weniger ausgeprägte Schwerpunktverteilung der Kompetenzen. Eine Konkretisierung der jeweiligen fachlichen Kompetenzzuschreibungen für jeden Prozessschritt erfolgt dahingehend, dass bezüglich der einzelnen Kompetenzschwerpunkte Empfehlungen für mögliche Kompetenzträger gegeben werden. Eine zusammenfassende Komplettübersicht bezüglich dieser Aspekte findet sich im Anhang.

Zudem wird für die Bearbeitung des Vorgehensmodells unterstellt, dass eine Projektleitung bestimmt wird. Ihre Aufgabe bei der praktischen Umsetzung dieses Vorgehensmodells ist es, die einzelnen Teilaufgaben an die entsprechenden Kompetenzträger zu verteilen, deren Ergebnisse einzufordern und zu kontrollieren sowie den Prozess der OS-Veröffentlichung zu überwachen und zu moderieren.

Die Verfasser hoffen auf vielfältige konstruktive Kritik der Leser und würden sich über Berichte über Anwendungserfahrungen freuen.

2 Schritt 1: Entscheidung für eine OS-Veröffentlichung

2.1 Ziele – Was soll durch diesen Schritt erreicht werden?

An der Spitze des OS-Vorgehensmodells steht die grundlegende strategische Entscheidung, ob die vorliegende Software in den Status einer OS-Software überführt werden soll. Deshalb muss diese Option mit möglichen Alternativen der OS-Veröffentlichung verglichen und die jeweiligen Vorteilhaftigkeiten gegeneinander abgewogen werden, um somit zu entscheiden, ob sich die OS-Veröffentlichung als beste Strategiemöglichkeit erweist.

2.2 Leitsätze und –fragen – Was muss beachtet werden?

Der eigentlichen Entscheidung gehen aber einige Arbeitsschritte voraus, dessen Ergebnisse später als Grundlage für die Entscheidung herangezogen werden. Diese Arbeitsschritte werden nachfolgend beschrieben.

Betreiben Sie eine Umfeldanalyse!

Ziel dieser Analyse ist es, das externe Umfeld der OSS veröffentlichende Institution daraufhin zu erkunden, ob dem Vorhaben wichtige Gründe im Wege stehen könnten oder ob sich besondere Chancen aus diesem Vorhaben ergeben. Dabei sollten sowohl allgemein gesellschaftliche, technische und politische Trends und Entwicklungen als auch sich im engeren Umfeld der Institution ergebene Risiken und Chancen analysiert werden. Diese ergeben sich aus den maßgeblichen Wettbewerbskräften, also beispielsweise Konkurrenten, Kunden, Zielgruppen etc..

Führen Sie eine Eigenanalyse durch!

Wichtig ist es zudem, die eigene Ressourcenkapazität für die Realisierung des Vorhabens zu kennen. Hier sollte geprüft werden, welchen strategischen Spielraum aus Ressourcensicht die Institution zur Realisierung des Vorhabens hat. Dabei sind sowohl personelle und sachliche als auch finanzielle Ressourcen gleichermaßen angesprochen.

Stellen Sie mögliche Gegenalternativen auf!

Die gewonnenen Informationen aus der Umwelt- und Eigenanalyse verdichten mögliche Strategiealternativen, die sich gegenüber der Möglichkeit einer OS-Veröffentlichung ergeben. Damit wird ein Raum für Strategien aufgezeigt. Aber auch die Chancen und Risiken der OS-Veröffentlichung werden durch die konkrete Auflistung und Benennung von Alternativstrategien der Softwaredistribution verdeutlicht.

Wählen Sie die strategisch günstigste Alternative!

„Aus dem aufgespannten Raum der Alternativen ist schließlich in einem Bewertungsprozess diejenige Strategie auszuwählen, die in Anbetracht der Stärken und Schwächen (…) und der zu erwartenden Bedrohungen und/oder Chancen aus der Umwelt den größten Erfolg verspricht“[1]. Dabei sind die möglichen Alternativen ganz sachlich mit dem Vorhaben der OS-Veröffentlichung zu vergleichen. Falls sich die Wahl zu Gunsten anderer Alternativen entscheiden sollte, so ist dies als Ergebnis ökonomisch-rationaler Betrachtungen hinzunehmen und das OS-Vorhaben entsprechend einzustellen.

2.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Der Prozessschritt der Entscheidungsfindung berührt alleine die betriebswirtschaftlichen Kompetenzen. Die hier zu fällenden strategischen Entscheidungen betreffen konkret das Management der höheren Ebenen einer Institution, weil eine derartige Strategieentscheidung von grundlegender Natur ist. Eine Aufwandseinschätzung gemäß der angestrebten Punkteverteilung ergibt sich deshalb wie folgt:

Technisch	
Rechtlich	
Betriebswirtschaftlich	●●●●●

Tabelle 1: Kompetenzverteilung für Schritt 1

[1] vgl. Steinmann/Schreyögg 2000, S. 158

2.4 Erläuterungen

Dem Prozess einer OS-Veröffentlichung geht immer eine Grundsatzentscheidung, ob die im Mittelpunkt stehende Software den Status einer OSS erhalten soll oder nicht, voran. Diese Entscheidung wird in dem hier behandelten strategischen Managementprozess gefällt[2]. In einem derartigen Managementprozess wird die Option der OS-Veröffentlichung auf der Grundlage einer eingehenden Umwelt- und Eigenanalyse, in denen die Umfeldbedingungen und eigenen Ressourcen auf das OS-Vorhaben abgestimmt wurden, mit anderen Alternativen der Softwarebereitstellung (zum Beispiel kommerzielle Vermarktung) verglichen. In einen anschließenden Auswahlprozess sollte sich schließlich die OS-Veröffentlichung als die günstigste Alternative erweisen

Im Folgenden werden mögliche Einflussfaktoren, die zu der OS-Entscheidung führen könnten, näher erläutert: Von öffentlichen Finanzmitteln abhängende Einrichtungen, wie Universitäten, Institute etc., entwickeln Software in der Regel im Rahmen ihrer Forschungsaktivitäten. Das Ergebnis dieser Forschungsarbeiten stellt ein mehr oder weniger marktreifes Softwareprodukt dar, dessen weitere Bestimmung nicht selten ungewiss ist. Neben anderen Alternativen stellt die Publizierung dieser Forschungsergebnisse eine sinnvolle und derzeit auch praktizierte Lösung dar. Diese Publikation kann beispielsweise dadurch erfolgen, dass die Software auf proprietärem Weg vertrieben wird. Für diese Möglichkeit sind allerdings gewisse Aufwände notwendig, für die in der Regel aber keine Budgets zur Verfügung stehen. Deshalb stellt die Strategie der OS-Veröffentlichung eine mögliche Alternative dar. Auf diese Weise kann der Aufwand des Publizierens für personelle, finanzielle und sachliche Ressourcen in überschaubaren Grenzen gehalten oder sogar gänzlich in die Hände von Dritten (zum Beispiel einer Community) übergeben werden. Somit stellt der Ressourcenaufwand für eine Softwarepublikation eine nicht unbedeutende Einflussgröße der Entscheidung für eine OS-Veröffentlichung dar.

Aber auch für privatwirtschaftliche Unternehmen stellt die Open Source-Strategie eine Möglichkeit dar. So kann aus einer Software, die unter Open Source-Lizenzen und -Bedingungen verbreitet wird, ein strategischer Nutzen gezogen werden. Die

[2] vgl. z.B. Steinmann/Schreyögg 2000, S. 157 ff.

Software kann beispielsweise auf breiterer Basis weiterentwickelt werden, wodurch frühzeitig ein Kontakt und damit ein Review durch potenzielle Anwender stattfindet, oder man etabliert sich in einem Themenfeld als Kompetenzträger, etc. Somit stellt sich für ein Unternehmen die Frage, entweder den proprietären oder den Open Source Weg zu gehen. Die Entscheidung für die Open Source Alternative hängt also von dem einfachen ökonomischen Sachverhalt ab, ob die Lizenzerträge aus der proprietären Alternative als niedriger prognostiziert werden können als die summierten indirekten Erträge aus einer OS-Strategie[3]. Bei dieser Bewertung spielen nicht ausschließlich monetäre Größen eine Rolle. Vielmehr ist es der insgesamt zu bewertende Nutzenertrag einer Alternative.

Die Entscheidung für die OS-Alternative gilt im Rahmen dieser Studie als Ausgangspunkt für den Prozess der OS-Veröffentlichung.

[3] vgl. Raymond 1999a

3 Schritt 2: Rechtliche Aspekte der SW klären

3.1 Ziele – Was soll durch diesen Schritt erreicht werden?

Aus der Software, die unter den Open Source Status gestellt werden soll, können sich rechtliche Aspekte ergeben, die der OS-Veröffentlichung im Wege stehen könnten und deshalb im Prozess einer OS-Veröffentlichung besonders zu berücksichtigen sind. Insbesondere Lizenzbedingungen sowie Eigentumsrechte können den Prozess behindern oder sogar gänzlich stoppen. Wegen dieser Brisanz sollten diese rechtlichen Fragen besondere Aufmerksamkeit geschenkt und in einem gesonderten Schritt behandelt werden.

Da die Analyse der rechtlichen Situation aber ergeben könnte, dass eine OS-Strategie nicht möglich ist, muss dem Prozess der OS-Veröffentlichung an dieser Stelle also eine Abbruchmöglichkeit eingeräumt werden.

3.2 Leitsätze und –fragen – Was muss beachtet werden?

Alle rechtlichen Aspekte, die die Software selbst oder ihr Umfeld betreffen, sind im Sinne einer IST-Analyse zu erfassen und zu dokumentieren. Die entsprechende Dokumentation dient dann den folgenden Schritten als Informationsgrundlage.

In diesen Arbeitsschritt fällt vor allem die Klärung von Lizenzbedingungen, die für Software-Toolkits, mit denen die Software hergestellt worden ist, oder für den integrierten Quellcode gelten. Zudem sind Verfügungs- und Urheberrechte der Software und Teilen davon, die mit Dritten (zum Beispiel Entwicklern oder Geldgebern) eingegangen worden sind bzw. die Dritte innehaben, zu berücksichtigen. Außerdem sind die Besonderheiten der Haftungs- und Gewährleistungsfunktionen, die sich im Zusammenhang mit der OS-Veröffentlichung ergeben, zu beachten. Es sind also gleichermaßen lizenzrechtliche, urheberrechtliche sowie haftungs- und gewährleistungsrechtliche Aspekte zu klären.

3.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Diese Aufgabe bedarf vor allem rechtliches Know-how und Hintergrundwissen und sollte dementsprechend von Experten auf diesem Gebiet erfüllt werden. So sollte diesbezüglich unbedingt der juristische Rat der Rechtsabteilung oder von

unabhängigen Rechtsanwälten eingeholt werden. Eine Punkteverteilung ergibt sich deshalb einheitlich zu Gunsten der rechtlichen Kompetenzen:

Technisch	
Rechtlich	●●●●●
Betriebswirtschaftlich	

Tabelle 2: Kompetenzverteilung für Schritt 2

3.4 Erläuterungen

3.4.1 Lizenzrecht

Um die Relevanz der lizenzrechtlichen Aspekte darzustellen, kann das Beispiel des Softwareherstellers Netscape genannt werden. So hatte Netscape bei der OS-Veröffentlichung seines Softwareproduktes Netscape Communicator vertragliche Verpflichtungen mit kooperierenden Unternehmen zu beachten, die besagten, dass der Communicator nur unter einer none-Open-Source Lizenz zur Verfügung zu stellen sei[4]. Auch sind eventuelle Lieferverbindlichkeiten mit Distributoren einzuhalten gewesen. Wie dieses Beispiel zeigt, ist insgesamt also darauf zu achten, dass durch die OS-Veröffentlichung nicht Verletzungen aus Lizenzvereinbarungen mit Dritten, die noch aus dem proprietären Lizenzstatus der Software stammen, entstehen.

Ähnliches gilt für das Verhältnis des Softwareherstellers zum Kunden. So kann die nun freie Verfügbarkeit der Software bei ehemaligen Kunden, die für dasselbe Produkt Geld bezahlen mussten, Verärgerung und Unverständnis auslösen. In diesem Fall kann das Softwareunternehmen schnell in Erklärungsnot geraten, wenn die ehemals gemachten kommerziellen Lizenzvereinbarungen nichtig werden oder von Seiten des Herstellers auf Grund der freien Verfügbarkeit nicht mehr eingehalten werden.

Für Software, die neu entwickelt und noch nicht veröffentlicht wurde, sind ebenfalls lizenzrechtliche Aspekte zu berücksichtigen. So kann es beispielsweise abhängig vom Lizenztyp eine bedeutende Rolle spielen, ob Softwareprodukte mit Open Source Toolkits programmiert wurden. Ist das der Fall, so sind gegebenenfalls die Open Source Lizenzbestimmungen des Toolkits für die spätere Klärung und Entwicklung

[4] vgl. Schaufler 2000

eigener lizenzrechtlicher Aspekte (vgl. Schritt 5) maßgeblich. So kann nämlich die Lizenz eines Toolkits bestimmen, dass mit dem Toolkit entstandene Produkte nur unter denselben Lizenzbedingungen verbreitet werden dürfen, die auch für das Toolkit gelten[5]. Derselbe Aspekt gilt für den Fall, wenn man bei der Entwicklung der Software Quellcode aus anderen (freien) Produkten übernimmt bzw. verwendet. Das betrifft Bibliotheken oder auch komplette Programmbestandteile. In diesem Fall sind gegebenenfalls die Lizenzbedingungen der Ursprungsversion zu berücksichtigen. Enthält z.B. die Lizenz des Urwerkes eine Werbeklausel, so könnte in den Lizenzbestimmungen festgelegt sein, dass auch das neue Softwareprodukt diese Werbeklausel enthalten soll.

3.4.2 Urheberrecht

Ferner sind generelle und eventuelle Urheberrechte von Autoren der Software abzuklären. Alles was nicht „public domain“ ist, unterliegt einem Copyright, möglicherweise auch mehreren. Gemäß der Berner Konvention (die seit 1978 US-Recht ist) muss das Urheberrecht nicht ausdrücklich erklärt werden. Das heißt: Der Urheber eines Werks besitzt das Urheberrecht daran, auch wenn das Werk nicht mit einer ausdrücklichen Copyright-Notiz versehen ist. Um dieses Recht nicht zu verletzen, ist vor der Open Source–Veröffentlichung zu klären, wer diese Rechte innehat. Wer als Urheber eines Werks gilt, kann ziemlich kompliziert sein, insbesondere bei Software, an der mehrere Personen gearbeitet haben.

Im Falle von nicht kommerziellen Softwareentwicklungen, beispielsweise im Rahmen von Forschungsprojekten, lässt sich die Urheberfrage relativ einfach beantworten, wenn eine einzelne Person die Software programmiert hat. In diesem Fall kann eben diese Person als Urheber festgemacht werden. In der Regel wird Software allerdings in Teams entwickelt. In diesem Fall sind die einzelnen Teammitglieder laut § 8 UrhG so genannte Miturheber, sofern ihre einzelnen Beiträge nicht gesondert verwertet werden können, sondern eben nur als Ganzes[6]. Das Urheberrecht der Software liegt dann bei allen Miturhebern gemeinsam[7].

[5] vgl. Siepmann 1999
[6] vgl. Schiffner 2003, S. 119
[7] vgl. § 8 UrhG Abs. 2

Auch bei der kommerziellen Entwicklung von proprietärer Software, die zumeist in Teams bestehend aus Angestellten eines Softwareunternehmens entsteht, kann zunächst nach § 8 UrhG von einzelnen Miturheberschaften der Teammitglieder ausgegangen werden[8]. Ist die Software aber von einer einzelnen Person entwickelt worden, so liegt die Urheberschaft bei eben dieser Person. In der Welt der kommerziellen Softwareentwicklung gehen in beiden Fällen die Vermögensrechte, sofern nichts anderes vereinbart ist, dann nach § 69b UrhG auf das Softwareunternehmen als Arbeitgeber über.[9]

Sofern das Urheberrecht nicht ohnehin bei der OSS veröffentlichenden Institution liegt, sind mit dem jeweiligen Inhaber des Urheberrechts dann entsprechende Übertragungen des Urheberrechts oder Genehmigungen für die weitere Nutzung der Software abzuklären. In der Praxis entscheidet man sich sogar häufig dafür, das Copyright auf die Free Software Foundation (FSF) zu übertragen. Dabei geht man davon aus, dass diese ein Interesse daran hat, Open Source Software zu verteidigen und ihren Status zu wahren.

3.4.3 Haftung und Gewährleistung

Haftungs- und Gewährleistungsaspekte im Zusammenhang mit der OS-Veröffentlichung sind ebenfalls von besonderer Relevanz. Die zentrale Frage für einen OSS-Anbieter ist dabei, ob man sich als Anbieter darauf verlassen kann, dass die als Open Source angebotene Software keinen Gewährleistungs- und Haftungsansprüchen durch die Abnehmer unterliegt. Dieser Fragestellung sind u. a. Jaeger (2000) sowie Werth (2003) nachgegangen, deren Überlegungen in dieser Arbeit Berücksichtigung finden sollen.

In nahezu jeder OS-Lizenz wird ein Gewähr- oder Haftungsanspruch von Seiten der Abnehmer der OSS ausgeschlossen. Die Lizenzbestimmungen besagen, dass die OSS nur auf eigene Verantwortung genutzt werden dürfen. Die Ziffern 11 und 12 der GNU Public License (GPL) bieten dafür ein schönes Beispiel[10].

[8] vgl. Schiffner 2003, S. 119
[9] vgl. Schiffner 2003, S. 119
[10] vgl. GPL 1991

> 11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTEND PERMITTED BY LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
>
> 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCUATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR ANY OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Diese Befreiung der OS-Anbieter von jeglichen Gewähr- und Haftungsverpflichtungen ist zwar im US amerikanischen Rechtsraum unbedenklich, aber wie sieht es mit der Wirksamkeit solcher Klauseln nach deutschem Recht aus? Das deutsche Recht lässt einen Haftungs- und Gewährleistungsausschluss nicht ohne weiteres zu.

Zunächst einmal sind die Begriffe Gewährleistung und Haftung voneinander zu trennen. Die Gewährleistung ist eine Form der vertraglichen Haftung dafür, dass eine Ware oder Dienstleistung ohne Mangel erbracht wird. Auf ein Verschulden kommt es dabei nicht an. Für Software bedeutet das nichts anderes als die ordnungsgemäße Funktionsfähigkeit des Programms. Die Haftung ist ein darüber hinaus reichender Begriff, der nicht nur bei Verträgen Anwendung findet. Im Prinzip haftet jeder für sein Tun und Unterlassen, in der Regel aber nur bei Vorsatz oder Fahrlässigkeit (also Verschulden). Die Haftung ist also angesprochen, wenn ein Programm Viren enthält - mag es sonst auch fehlerlos funktionieren.

Gewährleistung

Ziffer 11 der GPL betrifft die Gewährleistung und schließt diese komplett aus, jedenfalls "soweit gesetzlich zulässig".[11] Mit der GPL - dem Vertrag zwischen dem Nutzer und demjenigen, der das Programm vertreibt - sollen also die gesetzlichen Schutzvorschriften abgedungen werden. Das Problem daran: Die GPL stellt als vorformulierter Vertragsbestandteil eine "Allgemeine Geschäftsbedingung" (AGB) dar, die unter das AGB-Gesetz fällt. Dies dient dem Schutz der Verbraucher und soll verhindern, dass man sich mittels "Kleingedrucktem" jeglicher Haftung entziehen kann.[12] Dabei lässt das AGB-Gesetz anders als das US-Recht keine "geltungserhaltende Reduktion" zu. Das heißt, man kann mit der Formulierung "soweit gesetzlich zulässig" nicht die AGB bis zur gesetzlich zulässigen Grenze retten. Daher gelten die allgemeinen gesetzlichen Vorschriften, wenn die Klauseln gegen das AGB-Gesetz verstoßen - und genau das ist bei Ziffer 11 GPL der Fall. Denn ein kompletter Gewährleistungsausschluss ist unwirksam. Ziffer 11 GPL ist damit nicht mehr wert als gar kein Gewährleistungsausschluss. Dies bedeutet aber kein Ausschlusskriterium für die Verweigerung einer Gewährleistungsübernahme.

Für den Normalfall, in dem Software unter einer Open Source Lizenz kostenlos zum Download angeboten wird, kann man aber von einer "Schenkung" ausgehen. Verschenkt werden können nämlich nicht nur Sachen - § 516 des Bürgerlichen Gesetzbuches (BGB) spricht von einer "unentgeltlichen Zuwendung" - sondern auch andere Leistungen.[13] Das Nutzungsrecht an einem Programm ist eine solche Zuwendung. Und da derjenige, der etwas verschenkt, nicht über Gebühr für das Verschenkte verantwortlich sein soll, sieht das BGB nur eine Haftung für "arglistig verschwiegene Mängel" vor.[14] Wer also nicht weiß, dass sein Programm funktionsunfähig ist, der muss auch nicht für auftretende Mängel einstehen.

Komplizierter sieht die Sachlage aber für Distributoren aus: Wenn sie ein Programm auf CD-ROM und mit Handbuch oder Support zu einem Pauschalpreis vertreiben, ist eine Gewährleistung nach den Vorschriften des Kaufrechts denkbar, denn ein

[11] vgl. GPL 1991
[12] vgl. AGB 1976
[13] vgl. BGB 1998
[14] vgl. BGB 1998

Verkäufer unterliegt immer strengeren Bestimmungen als jemand, der etwas unentgeltlich leistet.

Haftungsausschluss

Für den Haftungsausschluss nach Ziffer 12 GPL gilt nichts anderes. Auch er ist nach den Vorschriften (§ 11 Nr. 7 AGB-Gesetz) des AGB-Gesetzes unwirksam.[15] Aber glücklicherweise sind die gesetzlichen Regelungen hier ebenfalls milde für den "Schenker". Er hat nach § 521 BGB nur Vorsatz und grobe Fahrlässigkeit zu vertreten.[16] Grobe Fahrlässigkeit tritt etwa dann ein, wenn jemand ein hastig zusammengefügtes Programm ohne weitere Prüfung anbietet und dann dadurch bei den Nutzern Schäden an Hard- oder Software auftreten.

[15]vgl. AGB 1976
[16]vgl. BGB 1998

4 Schritt 3: Zielsetzung

4.1 Ziele – Was soll durch diesen Schritt erreicht werden?

In diesem Schritt gilt es, die mit der OS-Veröffentlichung der Software verfolgten Ziele zu definieren und zu formulieren. Ziele geben dem Prozess der OS-Veröffentlichung eine gewisse Handlungsrichtung vor, anhand derer der Prozess verbindlich ausgerichtet und kontrolliert werden kann. Ohne eine zielorientierte Ausrichtung der OS-Veröffentlichung droht das Vorhaben zu einer reaktiven Anpassung an Umweltveränderungen zu degenerieren.

4.2 Leitsätze und –fragen – Was muss beachtet werden?

Konkrete inhaltliche Handlungsziele, zu denen auch die Ziele im Zusammenhang mit der OS-Veröffentlichung gezählt werden können, lassen sich in der Regel aus formulierten Strategien ableiten[17]. So sind die Ziele für die OS-Veröffentlichung eben aus dieser Strategie heraus, also der Entscheidung zur OS-Veröffentlichung der Software, abzuleiten. Auch die sich ergebene rechtliche Situation der betrachteten Software stellt eine Informationsbasis für die Formulierung dieser Handlungsziele dar. Dabei lassen sich Handlungsziele generell und die Handlungsziele in Bezug auf die OS-Veröffentlichung im besonderen in einer gewissen Rangordnung formulieren. Es lassen sich sowohl Ober- als auch Unterziele definieren, wobei die Oberziele die Definition der Unterziele prägen.

Ein weiterer bei der Zielformulierung zu beachtender Aspekt ist die Adäquanz der Ziele zu den generellen, übergeordneten Zielen der OSS veröffentlichenden Institution, die quasi als Leitlinien für die gesamte Institution dienen. So müssen sich die Ziele, die speziell mit der OS-Veröffentlichung verfolgt werden und als inhaltlich konkrete Handlungsziele angesehen werden können, in das hierarchische Zielsystem der Institution einfädeln lassen, um eine Konformität zu gewährleisten.

4.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Die Zielformulierung für die OS-Veröffentlichung wird in erster Linie von betriebswirtschaftlichen Kompetenzen bestimmt. Im Detail wird dies sehr wahrscheinlich vom strategischen Management der Institution in einer konkreten

[17] vgl. Schreyögg 1984, S. 87

Formulierung und Einordnung der Ziele in die Zielhierarchie bzw. das Zielsystem der Institution enden. Für die Formulierung der Ziele bedarf es aber auch eines technischen Hintergrundwissens. So muss von technischer Seite geklärt werden, ob potenzielle Ziele technisch umsetzbar sind. Eine Aufwandseinschätzung ergibt deshalb folgendes Bild:

Technisch	•
Rechtlich	
Betriebswirtschaftlich	••••

Tabelle 3: Kompetenzverteilung für Schritt 3

4.4 Erläuterungen

Zunächst stellt sich die Frage, wer (Uni/Unternehmen) die einzelnen Ziele verfolgt. Entsprechend können die Zielsetzungen, die mit der OS-Veröffentlichung einer Software verfolgt werden, von unterschiedlicher Natur sein. Generell lassen sich die Ziele meist nicht mit altruistischen Motiven begründen. Vielmehr sind es mehr oder weniger ausgeprägte strategische Ziele, die mit der OS-Veröffentlichung verfolgt werden.

An oberster Stelle einer möglichen Zielhierarchie steht dabei stets die Verbreitung der Software an eine möglichst breite Anwendergruppe. Dieses Ziel wird in der Regel durch die geringeren bis gar nicht aufzubringenden Anschaffungskosten für Open Source Software erreicht, die für den Nutzer einen nicht unerheblichen Anreiz darstellen. Die nachfolgend vorgestellten Ziele bedingen eine große Anwenderzahl einer Open Source Software und sind dem hier beschriebenen Ziel der Erreichung einer hohen Anwenderzahl untergeordnet.

Ein sehr populäres strategisches Ziel sieht vor, Einnahmen aus Zusatzgeschäften mit der Open Source Software zu erzielen. Zusatzleistungen zu einem Softwareprodukt können beispielsweise Beratungs-, Konfigurations-, Implementierungs- sowie Kompilierungsleistungen, aber auch Schulungen, Dokumentationen oder spezielle, unter proprieträrem Status angebotene Add-Ons etc. sein. Zwar sind dann keine Umsätze mehr durch den Kern der Leistung - also dem Softwareprodukt selbst - zu erzielen, allerdings können mit eben diesen Zusatzleistungen Umsätze generiert

werden. Das Softwareprodukt dient dann gewissermaßen als Köder zur Erzeugung von Nachfrage nach diesen Zusatzleistungen. Dem ursprünglichen Programmierer dieser Software kann in diesem Zusammenhang für die angebotenen Zusatzleistungen ein hohes, produktspezifisches Know-how bescheinigt werden, da er das Programm einst selbst entwickelt hat. Dies führt wiederum dazu, dass die Abnehmer dieser Zusatzleistungen dem Anbieter ein hohes Maß an Vertrauen entgegenbringen, was gerade für Zusatzleistungen mit hohem Dienstleistungsanteil von Bedeutung ist.

Ein weiterer Grund für die OS-Veröffentlichung einer proprietären Software kann auch in der Option gesehen werden, die eigene Position zu stärken. Dies trifft insbesondere für die erwähnten privatwirtschaftlichen Unternehmen, die ihre Position am Markt verteidigen müssen, zu. Verfügt in diesem Zusammenhang ein Unternehmen nur über einen geringen Marktanteil und droht vielleicht durch übermächtige Konkurrenten von diesem verdrängt zu werden, kann auch hier die Strategie der OS-Veröffentlichung hilfreich sein. Durch die freie Verfügbarkeit der Software nach der OS-Veröffentlichung kann – wie beschrieben - eine Vermehrung der Anwenderzahl und damit eine Erhöhung des Marktanteils vermutet werden. Dieser strategische Schritt würde zwar keine Umsatzsteigerung aus dem Verkauf der Software bescheren, aber es kann zumindest auf eine Verbesserung der Wettbewerbsposition gehofft werden. Zudem kann der Bekanntheitsgrad des Unternehmens gesteigert werden und auch das Image bzw. die Reputation könnte davon auf positive Weise profitieren. Dies könnte wiederum einen Einfluss auf den Absatz anderer Produkte und Leistungen eines Unternehmens haben. Der Aspekt der Image- und Reputationssteigerung kann aber auch für Universitäten wichtig sein, die mit dieser Strategie auf eine Dokumentation und Anerkennung ihrer Forschungsbemühungen hoffen können.

Ferner könnte eine Einrichtung, die eine OS-Veröffentlichung ihrer Software vollzieht, von einer kostenlosen Weiterentwicklung der Software profitieren. Dabei kann theoretisch auf einen schier unerschöpflichen und vor allem unentgeltlich zur Verfügung stehenden Pool von spezifischem Softwarewissen aus der Community zurückgegriffen werden.[18] Auch die Entwicklungs- bzw. Aktualisierungsgeschwindigkeit bei eventuell aufgetretenen Bugs ist in derartigen

[18]vgl. Raymond 1999b

Communities als hoch einzustufen. Die Frage des organisierten Bugfixing könnte auf diese Weise in die Hände von Communities gelegt werden.

Die Übergabe einer Software an eine Open Source Community würde auch eine Ressourcenschonung mit sich bringen. Dieser Aspekt ist für Software, die als Ergebnis öffentlicher Forschungsarbeiten (etwa an Universitäten o.ä.) hervor gegangen ist, von besonderer Bedeutung. So sieht der Budgetrahmen dieser Einrichtung eine kommerzielle Vermarktung dieser Software in der Regel nicht vor. Auf Grund dessen würde die Software in den Schubladen verstauben und der Öffentlichkeit vorenthalten. Die Strategie der OS-Veröffentlichfung würde hier Abhilfe leisten. Auch der einfache Gedanke die Ergebnisse aus Forschungsarbeiten und -projekten der Öffentlichkeit (unabhängig von verfügbaren Budgetrahmen oder Reputationsabsichten) zugänglich zu machen, kann ein Motiv darstellen, die OSS zu veröffentlichen.

5 Schritt 4: Formulierung eines Geschäftsmodells

Dieser Prozessschritt bezieht sich ganz konkret auf das im vorangegangenen Schritt festgelegte Ziel der Umsatzgenerierung durch Zusatzgeschäfte. Insofern ist dieser Schritt bei der Verfolgung anderer Ziele (vgl. Schritt 3) zu vernachlässigen und zu überspringen.

5.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Ziel dieses Schrittes ist es, ein Geschäftsmodell speziell für das Zusatzgeschäft um die OSS herum zu entwickeln. Damit ist gemeint, dass das Zusatzgeschäft hinsichtlich seiner Ausgestaltung zu konkretisieren ist und detailliert geplant werden soll. Die Ergebnisse der Konkretisierungs- und Planungsmaßnahmen sollten in einem Geschäftskonzept schriftlich festgehalten werden. Auf diese Weise wird ein Handlungsrahmen für die spätere Umsetzung des Geschäftsmodells (vgl. Abschnitt 13.2) aufgespannt, der den beteiligten Akteuren verbindliche Anhaltspunkte vorgibt.

Die Formulierung eines Geschäftsmodells bedarf dabei in der Regel sehr viel Detailtreue und ist deshalb sehr umfangreich. Aus diesem Grund kann diese Aufgabe im Rahmen dieses Kapitels nur angerissen und sehr oberflächig behandelt werden.

Besonders relevant ist auch der im Ablaufmodell frühe Zeitpunkt, an dem mit der Aufstellung des Geschäftsmodells begonnen werden sollte, denn dieser Arbeitsschritt benötigt aus oben genannten Gründen einen nicht unerheblichen Zeitaufwand und sollte bis zur Freigabe der OSS (vgl. Schritt 9) unbedingt abgeschlossen sein. Der Grund hierfür ist darin zu sehen, dass mit der Freigabe der Software unverzüglich mit der Vermarktung der Zusatzleistungen begonnen werden sollte, da anderenfalls möglichen Konkurrenten ein strategisches Zeitpolster für die Entwicklung eigener Geschäftsmodelle rund um die OSS geboten wird.

5.2 Leitsätze und –fragen – Was muss beachtet werden?

Die Aufstellung eines Geschäftsmodells für das Zusatzgeschäft soll sich an fünf Aspekten orientieren. Diese Aspekte entsprechen sinngemäß den Komponenten, die im Zusammenhang mit der Definition von Geschäftsmodellen stehen[19].

[19]vgl. zum Beispiel Bieger/Rüegg-Stürm/Rohr 2002, S. 36 ff.; Leiteritz 2002, S. 42 ff.

Die einzelnen Planungseinheiten, die für die Aufstellung eines Geschäftsmodells erforderlich sind, sollen im Folgenden anhand dieser fünf Aspekte kurz erläutert werden:

Markt

Die Zielgruppen für die Software und für die damit verbundenen Zusatzleistungen müssen definiert werden. Zudem sollten die geplanten Leistungen auf ihre Markttauglichkeit überprüft werden. Dabei sollten unter anderem folgende Detailfragen beantwortet werden:

- Wen möchte man mit der Veröffentlichung des Source Code erreichen?
- Gibt es bloß einen oder mehrere Märkte für die geplanten Leistungen?
  - Zielgruppe ausfindig machen
- Gibt es genügend potenzielle User, bzw. Entwickler für diese Software?
  - Anwenderbreite prüfen
- Wie möchte man sich am Markt präsentieren? (Branding)
- Wie intensiv müssen die Marketingmaßnahmen für die geplanten Angebote sein? (Spricht das Produkt für sich selbst?)
- Welche Marketingmaßnahmen sind für das Vorhaben die Richtigen?
  - Produkt-, Unternehmenswerbung, Mund-zu-Mund-Propaganda, etc...
- Welches sind die besten Distributions- bzw. Vertriebskanäle für diese Leistungen?
  - Internet, CD-ROM, Presse, direkte Kundenbetreuung, Handel, Partner

Gewinnmuster

Es muss vor der OS-Veröffentlichung geklärt werden, durch welche Zusatzleistungen rund um die Open Source Software in Zukunft Geld verdient werden soll. Diese Zusatzleistungen müssen entsprechend definiert und in einem weiteren Schritt ausgestaltet werden, ein Preis für diese Leistungen muss ermittelt werden etc. Beispielsweise müssen folgende Fragen beantwortet werden:

- Mit welchen Leistungsarten zur Software lässt sich Geld verdienen?
  - ⇒ Dienstleistungen: Service, Training, Implementationen, etc.
  - ⇒ Sachleistungen: Datenträger, Bücher, Dokumentationen, etc.

- Welche Einnahmestrukturen haben die Leistungen?
  - Stundensätze für Dienstleistungen, Stückpreise für Produkte, langfristige Serviceverträge, etc.
- Gibt es mehrere Gewinnmodelle?

Ressourcenfokus

Hierbei gilt es einen Plan aufzustellen, wie viele und welche Ressourcen für das Geschäftsmodell eingeplant werden müssen. Dabei sind sowohl finanzielle und personelle als auch materielle Ressourcen zu berücksichtigen.

- Welche Ressourcen steckt das Unternehmen / die Organisation in das Geschäftsmodell?
  - Geld, Personal, Hardware, Zeit, etc.
  - Welche Kompetenzen brauchen die eingebundenen Mitarbeiter?
- Auf welche Leistungen/Dienste/Produkte wird der Schwerpunkt gelegt?
- Gibt es Partner, die bei der Vermarktung der Zusatzleistungen mitwirken könnten?

Geschäftsstrategie

Unter diesem Stichwort wird erörtert, wie man Kunden für das Zusatzgeschäft gewinnen und auf Dauer halten kann, wie man der Konkurrenz entgegentritt, Bedrohungspotenziale ausschließt und sich vor Nachahmern schützt. Beispielsweise sind folgende Fragen zu berücksichtigen:

- Wie lassen sich die Kunden am besten gewinnen und an das Unternehmen binden?
  - Die eigene Kompetenz herausstellen (Know-How-Vorsprung durch eigene Entwicklung der freigestellten Software)
  - Durch eigene Leistungen eine Vertrauensbasis schaffen
- Wie sind die Machtverhältnisse zwischen Anbieter und Kunde?
- Wie hoch ist die Preissensitivität der Kunden?
- Wie ist die Konkurrenz in dem Bereich? / Wie hoch sind die Markteintrittsbarrieren?

  - Haben sich einige Unternehmen in dem Bereich schon etabliert und einen festen Kundenstamm, ist es für einen Neuling schwierig eine Nische zu finden.
  - Meist verursacht durch die freie Verfügbarkeit der OS-Software, ist es für Konkurrenten mit ähnlichem oder sogar gleichem Angebot relativ einfach in den Markt einzutreten.
- Wo liegen Bedrohungspotenziale?
  - Der Markt für OSS könnte generell bedroht werden, wenn es Herstellern von proprietärer Software (z.B. Microsoft) gelänge, durch eine andere Marketingpolitik und bessere Produkte Marktanteile zu gewinnen.
  - Wenn die Software generell immer einfacher zu bedienen und zu konfigurieren wird, benötigen die Kunden keine Dienstleistungen mehr und den Unternehmen fehlen die Kunden.
- Wie kann man sich vor Nachahmern schützen?
  - Das ist generell schwierig, da Zusatzleistungen in diesem Bereich rechtlich nur schwer schützbar sind (beispielsweise durch Patente). Daher müssen Alleinstellungsmerkmale aufgebaut werden, um sich so von Konkurrenten abzugrenzen.

Unternehmenskultur / Organisation / Mitarbeiter

Hierbei geht es darum, die richtigen Mitarbeiter einzubinden bzw. bestehende Mitarbeiter vom Geschäftsmodell zu überzeugen. Diese müssen über die notwendigen Einstellungen verfügen. Dabei ist insbesondere der mitarbeiterseitige Umgang mit der OS-Gemeinde zu erwähnen. Die Einstellungen der Mitarbeiter sollten sich auch in der Kultur der betreffenden Institution widerspiegeln. Zudem ist eine entsprechende Anpassung des Organisationssystems der Institution auf das Geschäftsmodell relevant. Insgesamt geht es bei diesem Aspekt um die Herausforderung, die entsprechenden internen Prozesse für den Erfolg des Geschäftsmodells anzustoßen. Dabei sind beispielsweise folgende Fragen zu beantworten:

- Welche Organisationsform ist für das Geschäftsmodell sinnvoll?
- Sind die eingebunden Mitarbeitern von der Open Source-Idee sowie dem Geschäftsmodell überzeugt?

5.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

In diesen Prozessschritt wird vor allem die strategische Planungseinheit der entsprechenden Institution involviert sein. Diese liegt meist beim Management der mittleren bis oberen Führungsebene. Wie die Ausführungen gezeigt haben, sind dafür alleine betriebswirtschaftliche Kompetenzen gefragt. Neben den organisatorischen Tätigkeiten, wie z.B. die Erarbeitung der Geschäftsstrategie, die Bestimmung der Zielgruppe und dem Erstellen von Zusatzangeboten, spielt hier aber auch der Finanzbereich eine Rolle. Seine Aufgabe ist es, einen Finanzierungsplan für das Vorhaben zu erarbeiten. Eine Punkteverteilung ergibt sich zu Gunsten der betriebswirtschaftlichen Kompetenzen.

Technisch	
Rechtlich	
Betriebswirtschaftlich	●●●●●

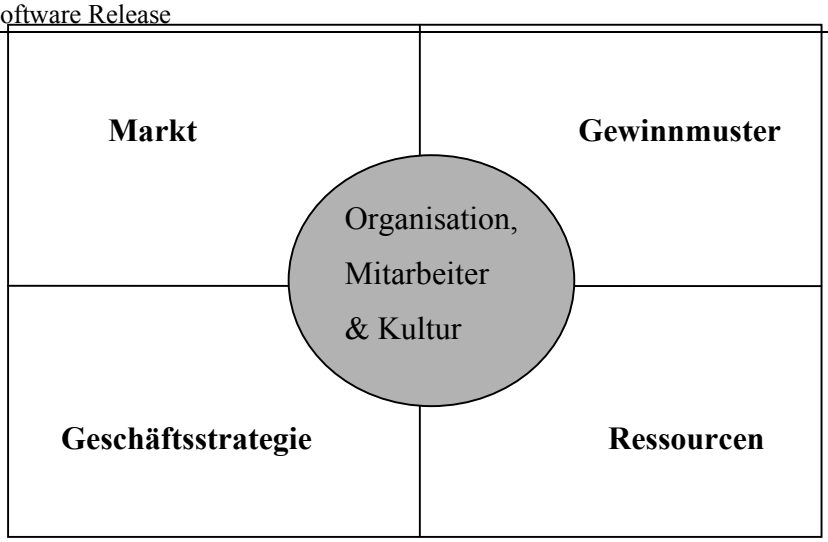
Tabelle 4: Kompetenzverteilung für Schritt 4

5.4 Erläuterungen

Die OSS veröffentlichenden Institutionen, die in diesem Kapitel im Mittelpunkt der Betrachtung stehen, verfolgen die OS-Veröffentlichung als strategisches Ziel (vgl. Kapitel 4). Ihr Ziel ist es, ein Geschäft aus Komplementär- und Zusatzprodukten rund um die OSS aufzubauen. Für dieses Zusatzgeschäft muss deshalb ein Geschäftsmodell aufgestellt werden.

In der Literatur gibt es keine einheitliche oder fest etablierte Definition eines Geschäftsmodells. Vielmehr ist ein Flickenteppich an Definitionen und Ansätzen zu beobachten. Zudem beschränken sich einzelne Definitionsversuche auf bestimmte Unternehmenstypen oder –branchen. In der vorliegenden Arbeit wurden die Komponenten des Definitionsansatzes von Nehls/Baumgartner (2000) herangezogen und für den vorliegenden Untersuchungsgegenstand entsprechend angepasst (vgl. Abbildung 1).

Abbildung 1: Komponenten zur Ausgestaltung eines Geschäftsmodells Quelle: Nehls/Baumgartner 2000



Mit der Hilfe dieser Komponenten soll nun die konkrete Ausgestaltung des Geschäftsmodells erfolgen. Nachfolgend soll anhand der einzelnen Aspekte die Formulierung des Geschäftsmodells detaillierter erläutert werden.

Markt

Unter diesem Aspekt wird die Beziehung der OS-veröffentlichenden Institution zum Markt beschrieben. Dabei wird sowohl die Beziehung, die sich sowohl aus dem Kernprodukt (OSS) als auch aus dem Zusatzgeschäft ergeben, betrachtet. Insbesondere kommt es hier auf die Identifizierung von Zielkunden und Marktsegmenten für die Leistungen aus dem Zusatzgeschäft an. Auch die Festlegung eines Preises für die Zusatz- bzw. Komplementärprodukte fällt unter diesen Aspekt. Neben der Preisermittlung sind hier noch weitere Marketingaktivitäten von Interesse. Beispielsweise ist aus distributionspolitischer Perspektive die Frage zu klären, welche Absatzwege und –kanäle für die OSS und die Zusatzleistungen zu wählen sind. Von kommunikationspolitischer Bedeutung ist der Marktauftritt der Institution. Hierunter fallen Werbemaßnahmen etc. Auch zur Markenpolitik der Produkte sollte sich entsprechend Gedanken gemacht werden.

Gewinnmuster

Ein bedeutender Punkt in der Ausarbeitung der OS-Strategie ist die Frage, mit welchen Leistungen die Institution Geld verdienen will. Hinter diesem Aspekt verbirgt sich also das Design der Komplementär- bzw. Zusatzleistungen. Marketingpolitisch geht es also um das Produktmanagement. Dabei sind mit Leistungen sowohl Produkte als auch Dienstleistungen gemeint. In der Literatur und in der Praxis werden verschiedene Möglichkeiten für ein Zusatzgeschäft diskutiert,

die sich nach der Beschaffenheit in Dienstleistungen oder Produkte einteilen lassen. Die folgende Tabelle 5 gibt einen Überblick und Ansatzmöglichkeiten.

Produkt	Proprietäre Software	Unter dem Status einer Proprietären Software wird Zusatzsoftware für die OSS angeboten. (z.B. Add-Ons)
	Accessories	Verkauf von Produkten, die mit der OSS zusammenhängen (Handbücher, CD-Roms, Gimmicks etc.)
Dienstleistungen	IT-Consulting	Es werden Dienstleistungen rund um die OSS angeboten (z.B. Beratungen Installationen Integrationen, Support etc.) z.B. Unisys, Accenture Alcove Linux Partner
	Schulungen	Schulungsangebote auf der Basis der OSS
	Mediatorenservice	Betreiben einer Internetplattform, über die die verschiedenen Interessensgruppen (Entwickler, Nutzer,, Dienstleister, Werbetreibende, Sponsoren) der OSS zusammengebracht werden. Über diese können beispielsweise Entwickler die Software schreiben, ablegen, verwalten oder kompilieren. Ferner können die Entwickler miteinander Kommunizieren. Nutzer können beispielsweise Softwarepakete, Anleitungen oder Dokumentationen downloaden. Werbetreibende können hier zielgerichtet Werbung platzieren. OSS Dienstleister können zielgerichtet ihre Dienstleistungen vermarkten

Tabelle 5: Mögliche Zusatz- und Komplementärleistungen Quelle: Eigene Darstellung

Ressourcenfokus

Bei diesem Aspekt geht es um die Entscheidung, welche und wie viele Ressourcen für das Geschäftsmodell erforderlich sind. Im Vordergrund steht dabei immer die Frage, ob entsprechende Ressourcen in ausreichender Menge und Qualifikation zur Verfügung stehen. Neben personellen Ressourcen müssen dabei auch die sachlichen und finanziellen Ressourcen berücksichtigt werden. Stehen irgendwelche Ressourcen nicht zur Verfügung, so muss überlegt werden, ob diese in einem ökonomischen Kosten-/Nutzenverhältnis beschaffen werden können. Anderenfalls könnten fehlende Ressourcen ein Ko-Kriterium für die Auswahl bestimmter Leistungen darstellen.

Geschäftsstrategie

Unter dem Aspekt der Geschäftsstrategie sind Maßnahmen in Betracht zu ziehen, die das Angebot des Zusatzgeschäfts gegenüber Wettbewerbern absichern. Gerade auf dem Markt für Softwareprodukte und –dienstleistungen sind sich die Angebote der verschiedenen Anbieter oft sehr ähnlich. Hinzu kommt, dass sich spezifisches Know-how patentrechtlich nicht schützen lässt. Die Markteintrittsbarrieren für diesen Markt sind deshalb sehr gering. Dies führt dazu, dass sich einzelne Anbieter mit ihrem Angebot nur schwer von ihren Konkurrenten abheben können, was wiederum sehr oft in Preiskämpfen endet. Durch strategische Maßnahmen kann hier interveniert werden. Beispielsweise stellen Markteintrittsbarrieren, Alleinstellungsmerkmale und die Nachhaltigkeit des Angebotes entsprechende strategische Maßnahmen dar, das Angebot zu individualisieren. Beispielsweise lassen sich durch spezialisierte Kenntnisse und Erfahrungen in einem bestimmten Bereich Alleinstellungsmerkmale aufbauen.

Organisation, Mitarbeiter und Kultur

Unter diesem Aspekt ist die Abstimmung der internen Prozesse der OSS veröffentlichenden Institution auf das Geschäftsmodell gemeint. So ist als erstes ein adäquates Organisationssystem aufzustellen bzw. das bestehende Organisationssystem der OSS veröffentlichenden Institution entsprechend anzupassen. Davon betroffen sind die Führung, das Personalmanagement, die Organisationsform sowie die Infrastruktur der Institution.

Auch die Mitarbeiter sowie die Kultur der Institution sind auf die neue Situation entsprechend einzustellen. So muss beispielsweise eine Forschungseinrichtung, die ihre Bemühungen bisher alleine auf die Entwicklung von Software fokussiert hat und sich nun ein Standbein durch den Verkauf von Zusatzleistungen aufbauen will, auf die Situation einer kommerziell handelnden Unternehmung eingeschworen werden. Umgekehrt müssen die Mitarbeiter kommerzieller Unternehmen, das Vorhaben der nun kostenlosen Weitergabe der einstigen proprietären Software verinnerlichen. Derartige Umstellungen sind durch ein geschicktes Management der Kultur und der Mitarbeiterführung einer Institution zu bewerkstelligen.

6 Schritt 5: Lizenzmodell wählen

Nach der Konkretisierung der OS-Strategie kann die Auswahl eines geeigneten OS-Lizenzmodells beginnen. Die Entscheidungsgrundlage für die Wahl eines Lizenzmodells sind die Ergebnisse der vorangegangenen Schritte.

6.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Ziel dieses Schrittes ist es, ein geeignetes Lizenzmodell für die im Mittelpunkt stehende OSS zu finden. Dabei sollte das Lizenzmodell so gewählt werden, dass es der OS-Strategie dienlich ist.

6.2 Leitsätze und –fragen – Was muss beachtet werden?

Für die Wahl des Lizenzmodells bieten sich grundsätzlich zwei Möglichkeiten. Einerseits kann aus vorhandenen Lizenzen anderer OSS eine für die eigenen Bedürfnisse weitgehend mit den eigenen Bedürfnissen übereinstimmende Lizenz, die gewissermaßen als Muster dient, ausgewählt werden. Diese kann dann gegebenenfalls weiter angepasst werden. Andererseits kann ein gänzlich neues Lizenzmodell definiert werden. Diese Möglichkeit bedarf allerdings eines bedeutenden Mehraufwands. Zudem erfüllen bestehende OS-Lizenzen in der Regel bestimmte Formvorschriften und formelle Standards, die zwar nicht verpflichtend sind, aber in OS-Kreisen gerne gesehen werden.

Als Informationsgrundlage für die Aufstellung einer Lizenz sollte die entsprechende Dokumentation aus Schritt 2 herangezogen werden, in der die zu berücksichtigenden lizenzrechtlichen Aspekte aufgelistet sind.

Zudem sollten bei der Lizenzwahl einige Handlungsempfehlungen beherzigt werden. So sollte die Lizenz unbedingt den Anforderungen der Open Source Software Initiative (OSI) entsprechen. Wichtig ist außerdem, die Kompatibilität der Lizenz mit Lizenzen aus dritter Software zu berücksichtigen, beispielsweise wenn Quellcode aus fremder Software in die eigene Software integriert oder mit entsprechenden Software-Toolkits programmiert wurde. Auch sollten sich selbst keine gravierenden Sonderrechte eingeräumt werden, die die Community verärgern könnten und somit

eine Zusammenarbeit erschweren würde. Der Einfachheit halber sollte zudem möglichst auf vorhandene Lizenzmodelle zurückgegriffen werden.

6.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Die lizenzrechtlichen Fragen dieses Prozessschrittes dürften in erster Linie die rechtlichen Kompetenzen fordern. Wie die obigen Ausführungen gezeigt haben und die folgenden Erläuterungen bestätigen werden, ist bei der Formulierung der Lizenzen auf die Daten der Zielsetzung und Strategieformulierung zurückzugreifen. Deshalb sind für diesen Schritt auch die betriebswirtschaftlichen Kompetenzen gefragt. Eine Punkteverteilung sieht entsprechend wie folgt aus:

Technisch	
Rechtlich	••••
Betriebswirtschaftlich	•

Tabelle 6: Kompetenzverteilung für Schritt 5

6.4 Erläuterungen

6.4.1 Die Open Source Lizenz

In Softwarelizenzen werden im Allgemeinen drei wesentliche Rechtsfragen behandelt.

Dies sind das Recht zum Vervielfältigen und Weiterverbreiten, zum Benutzen, zum Verändern für den privaten Gebrauch und zum Verbreiten der veränderten Kopien[20].

Auch OS-Lizenzen greifen diese Rechtsfragen auf. Die Open Source Initiative (OSI) regelt in einer offiziellen Definition in welcher Form diese Rechtsfragen für OSS ausgestaltet sein müssen[21]. Dazu entwickelte die OSI einen Kriterienkatalog, dessen Elemente eine OS-Lizenz beschreiben. Die wesentlichen Punkte aus diesem 10-Punkte-Katalog sind dabei die folgenden:

- Ein unbegrenztes Recht der Vervielfältigung muss bewilligt sein.
- Ein unbegrenztes Recht der Benutzung muss bewilligt sein.
- Ein unbegrenztes Recht der Veränderung für den privaten Gebrauch muss bewilligt sein.

[20] vgl. Raymond/Lohmeier 2001
[21] vgl. OSD 2003

„Die Richtlinien untersagen Einschränkungen bei der Weitergabe von geänderter binärer Software; das betrifft die Bedürfnisse von Software-Distributoren, die in der Lage sein müssen, laufenden Code ohne Belastung zu vertreiben. Es gestattet Urhebern zu verlangen, dass veränderter Quellcode als originaler Quellcode plus Patches weiterzugeben ist, um so die Absichten des Urhebers zu verdeutlichen und eine Übersicht aller Änderungen für Andere zu bieten.“[22]

Die einzelnen Punkte der Open Source Definition (OSD) stellen die offiziell anerkannten Bedingungen für eine OSS-Lizenz dar. In dieser Eigenschaft stellt sie zugleich eine Definition von Open Source-Software dar.

6.4.2 Verschiedene OS-Lizenzen

Kategorisierung

Die Lizenzen können entsprechend ihrem lizenzrechtlichem Charakteristika in fünf verschiedene Gruppen eingeteilt werde[23]

- Lizenzen ohne Copyleft-Effekt
- Lizenzen mit strengem Copyleft-Effekt
- Lizenzen mit beschränktem Copyleft-Effekt
- Lizenzen mit Wahlmöglichkeiten
- Lizenzen mit Sonderrechten

Im Folgenden werden die Lizenzarten näher erläutert und jeweils Beispiel-Lizenzen genannt. Die Kategorisierung wurde dem Lizenz Center (o.J.) entnommen.

Lizenzen ohne Copyleft-Effekt

Lizenzen ohne Copyleft-Effekt zeichnen sich dadurch aus, dass sie dem Lizenznehmer alle Freiheiten einer Open Source Lizenz einräumen und für Veränderungen der Software keine Bedingungen hinsichtlich des zu verwendenden Lizenztyps enthalten. Damit kann der Lizenznehmer veränderte Versionen der Software unter beliebigen Lizenzbedingungen weiterverbreiten, also auch in proprietäre Software überführen.

[22] vgl. Raymond/Lohmeier 2001
[23] Lizenz Center o.J.

Beispiellizenzen dafür sind:

- BSD License (original)
- Apache License (v. 1.1)
- MIT License
- OpenLDAP Public License (v. 2.7)
- Zope Public License (v. 2.0)

Lizenzen mit strengem Copyleft-Effekt

Bei Lizenzen mit einem strengen Copyleft-Effekt wird der Lizenznehmer verpflichtet, von der ursprünglichen Software abgeleitete Werke ebenfalls nur unter den Bedingungen der Ursprungslizenz weiterzuverbreiten. Die hier aufgeführten Lizenzen sind deswegen aber nicht schon unbedingt "GPL-kompatibel".

Beispiellizenzen dafür sind:

- GNU General Public License (GPL) (v.2.0)
- IBM Public License
- Common Public License
- Red Hat eCos Public License (v. 1.1)
- Open Software License

Lizenzen mit beschränktem Copyleft-Effekt

Lizenzen mit beschränktem Copyleft-Effekt gleichen denen mit strengem Copyleft-Effekt. Sofern Modifikationen der Software in eigenen Dateien realisiert werden, können diese Dateien auch unter anderen, z.B. proprietären Lizenzbedingungen, weiterverbreitet werden. Damit soll die Kombination von Software unter verschiedenen Lizenztypen erleichtert werden.

Beispiellizenzen dafür sind:

- Mozilla Public License
- GNU Lesser General Public License (LGPL) (v. 2.1)
- Nokia Open Source License
- Sun Public License
- Interbase Public License

Lizenzen mit Wahlmöglichkeiten

Diese Lizenzen sehen unterschiedliche rechtliche Folgen vor, je nachdem wie umfangreich eine Modifikation ist. Zudem werden dem Lizenznehmer verschiedene Wahlmöglichkeiten eingeräumt, wie Weiterentwicklungen weiterverbreitet werden können.

Eine Beispiellizenz dafür ist:

- Artistic License (v. 2.0)

Lizenzen mit Sonderrechten

Die Lizenzen mit Sonderrechten gewähren den Lizenznehmern zwar alle diejenigen Rechte, die Freie Software ausmachen, sehen aber zugleich besondere Privilegien für den Lizenzgeber bei Weiterentwicklungen durch den Lizenznehmer vor. Diese Lizenzen werden zumeist bei Programmen verwendet, die ursprünglich proprietär vertrieben wurden.

Beispiellizenzen dafür sind:

- Netscape Public License (NPL) (v. 1.1)
- Q Public License (QPL)
- Apple Public Source License (v. 1.2)
- OCLC Research Public License (v. 2.0)

Die GNU General Public License (GPL)

Die GNU General Public License (GPL) ist sicherlich die wichtigste und weitverbreitetste Open Source Lizenz. Mit dieser Lizenz fing der Gedanke von freier Software an, eine rechtliche Absicherung zu erhalten. Geschaffen wurde sie 1984 von Richard Stallmann als Lizenz für die Software des GNU Projekts. Sie enthält nicht nur die Bestimmungen zur Benutzung der Software, sondern auch eine lange Präambel, die den Hintergrund freier Software erklärt. Ein wichtiger Punkt der GPL, der allen anderen Open Source Lizenzen fehlt, ist die Idee des "Copyleft". Hierbei wird durch Ausnutzung des Copyrights dafür gesorgt, dass ein Programm, welches unter der GPL steht, auch frei bleibt. Dies wird dadurch erreicht, dass jedes Programm, welches eine Ableitung eines GPL-Programmes ist, automatisch wieder unter die GPL fällt. Zu beachten ist auch, dass es die GPL - im Gegensatz zu einigen anderen OSS-Lizenzen - nicht verbietet, mit Software unter dieser Lizenz Geld zu

verdienen. Es ist also durchaus erlaubt, auch deutlich mehr als nur die Kopierkosten zu verlangen, solange die Bedingungen der Lizenz erfüllt werden. Einen Aspekt des Selbstschutzes bietet die GPL dadurch, dass ein Verstoß gegen eine der Auflagen der Lizenz automatisch zum Verlust dieser Lizenz für die entsprechende Software führt.[24]

Die GPL hat sozusagen politische Ziele, da sie neben den schon bekannten Haftungsausschlüssen auch noch die Forderung enthält, dass alle Weiterentwicklungen und alle Programme, die in irgendeiner Form unter der GPL lizenzierten Code enthalten, wiederum unter der GPL veröffentlicht werden müssen. Diese Eigenschaft hat ihr den Spitznamen GNU „GNU is not Unix“ eingetragen. Dahinter steht nicht nur der Nutzen für die freie Software, sondern auch eine politische Motivation im Umgang mit dem Copyright. Aus Sicht der GPL verhindert das Copyright den natürlichen Umgang mit anderen Computernutzern, da es einigen Privilegien einräumt (denjenigen, die eine Software lizenziert haben). Ferner verbietet die GPL auch das Hinzufügen weiterer Einschränkungen, die sich auf den Quellcode beziehen. Diese Einschränkung macht GNU-Software, insbesondere Bibliotheken, im Wesentlichen für eine Verwendung im kommerziellen Umfeld unbrauchbar.

Die GNU Library General Public License (LGPL)

Die LGPL entspricht im wesentlichen der GPL, mit der Ausnahme, dass Programme, die lediglich mit einer der LGPL unterliegenden Bibliothek verbunden werden, nicht als abgeleitete Arbeit im Sinne der GPL betrachtet werden. So wird die Verwendung von durch LGPL geschützten Bibliotheken zur Entwicklung kommerzieller bzw. nicht der GPL unterliegender Software ermöglicht. Änderungen an der Bibliothek selbst müssen allerdings wieder der LGPL unterliegen. Software, die der LGPL unterliegt, bietet damit bessere Möglichkeiten für eine Verbindung zwischen freien und kommerziellen Entwicklungen.[25]

Die BSD License

Die BSD Lizenz ist eine der ältesten Lizenzen und schränkt sowohl Programmierer als auch Anwender nur insofern ein, dass die ursprünglichen Autoren genannt werden müssen. Hauptpunkt der Lizenz ist der Ausschluss von Haftungsansprüchen

[24] vgl. Marx 2001, S. 49
[25] vgl. o.V. 1999, S. 14

gegenüber den Programmierern. Sourcecode, der der BSD Lizenz unterliegt, kann in eigenen Entwicklungen benutzt werden, ohne dass diese wiederum freie Software sein müssen. Die BSD Lizenz stammt von der Universität von Berkeley. BSD steht für Berkeley Software Distribution.[26]
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, die Anfang der 1990er als Open-Source-Software zur Verfügung gestellt wurde (daraus entwickelten sich verschiedene freie Unices: OpenBSD, NetBSD, FreeBSD usw.). Die Lizenzbedingungen sind einfach:

- Das Programm darf in jeder Form, auch in Binärform, weitergegeben werden. Eine Pflicht zur Überlassung des Quellcodes besteht nicht.
- Bei der Weitergabe in Binärform muss die Lizenz den Dateien beigefügt werden.
- Bei Derivaten darf der Name der Autoren/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 aufgrund von Beschwerden 1999 entfernt.

Die NPL (Netscape Public License) und MPL (Mozilla Public License) Lizenzen
Die Netscape Public Licence (NPL) ist die Lizenz, unter der am 1.April 1998 der Quelltext des Netscape Communicators veröffentlicht wurde. Eric Raymond und Bruce Perens von der Open Source Initiative halfen, die einzelnen Paragraphen der Lizenz zu formulieren, aber die zentralen Aspekte steuerte Netscape selber bei.
Teile des Codes, insbesondere die später entwickelten, unterliegen einer zweiten Lizenz, der Mozilla Public Licence (MPL). Sie unterscheidet sich nur insofern von der NPL, dass sie keine Klauseln enthält, die Netscape besondere Rechte gewähren, wie zum Beispiel die Nutzung von Erweiterungen in anderen, nicht-freien Netscape-Produkten, und sich nicht im Besonderen auf den Netscape Communicator bezieht.

[26] vgl. o.V. 1999, S. 11

Modifikationen des vorhandenen Codes müssen wiederum der NPL/MPL unterliegen, wohingegen Erweiterungen (neue Routinen in separaten Dateien) ein anderes Copyright (oder gar keines) tragen können.
Obwohl die NPL im Falle Netscapes auf eine spezielle Geschäftssituation zugeschnitten wurde, nehmen einige Unternehmen sie oder eine ihrer Ableitungen als Lizenz für ihre Produkte, die sie im Zuge der Open-Source-Euphorie veröffentlichen. Schließlich handelt es sich beim Netscape Communicator um ein ehemals proprietäres Produkt, dessen Umwandlung in freie Software einer besonderen Lizenz bedurfte.[27]
Bei MPL wird unterschieden zwischen Datei-Derivaten und Werkderivaten. Datei-Derivate 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 Source-Code von Datei-Derivaten muss auf Anfrage ausgehändigt werden; Datei-Derivate 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 bei einem Web-Browser nachvollziehbar, aber auch für andere modulare Systeme geeignet.
Zwar schreibt die MPL für Datei-Derivate die Aushändigung des Quellcodes 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 salvatorische Klausel, was bedeutet, dass die verbleibenden Bedingungen auch rechtskräftig sind, wenn es einzelne Bedingungen auf Grund im Lande des Rechtsvorgangs geltender Bestimmungen nicht sind. So könnten z.B. Kryptographie-Exportverbote eine Verbreitung bestimmten Quellcodes verhindern, dies würde jedoch die übrigen MPL-Bedingungen nicht tangieren, sie müssen soweit wie möglich erfüllt werden.

[27] vgl. o.V. 1999, S. 15

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 derartig designierter Dateien mit GPL-Code ermöglicht.[28]

Die Apache Software License

Die Apache Software Foundation, die von namhaften IT-Unternehmen unterstützt wird, hat für ihre Open-Source-Serverprodukte (am bekanntesten davon der Apache Webserver) eine eigene Lizenz entwickelt. Diese Lizenz erlaubt Veränderungen am Quellcode und die ausschließliche Weitergabe in Binärform, sofern die Lizenzbedingungen 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 License ist sehr stark auf Apache-Produkte spezialisiert und empfiehlt sich kaum für allgemeine Open-Source-Applikationen.[29]

Die Artistic License

Die Artistic License wird für die Programmiersprache Perl verwendet. Perl selbst ist parallel unter der GPL lizenziert, das gleiche gilt für viele Perl-Programme und Bibliotheken. Aufgrund von juristischen Unklarheiten bei bestimmten Formulierungen wurde die Artistic License durch die Clarified Artistic License abgelöst.

Die Artistic License gestattet kostenlose Verbreitung und die Weitergabe von Veränderungen, unterscheidet dabei aber zwischen der sog. Standard-Version und Derivaten von dieser. Sie erlaubt es, proprietäre Derivate zu erstellen, welche aber eindeutig gekennzeichnet sein müssen (andere Dateinamen, Dokumentation der Unterschiede). Die freie Bereitstellung des Quellcodes eigener Veränderungen kann einmalig erfolgen, z.B. über das Posting im Usenet oder den Upload in ein öffentliches Datei-Archiv. Es dürfen Distributionsgebühren gefordert werden (also Gebühren für die bloße Bereitstellung der Software), aber keine Lizenzgebühren (also Gebühren, die z.B. an die Zahl der Nutzer gebunden sind). Die Vielzahl von

[28] vgl. o.V. 2003c
[29] vgl. o.V. 2003c

Fallunterscheidungen macht die Artistic License zu einer recht komplizierten Lizenz.[30]

Die QPL und andere Lizenzen

Die QPL und andere Lizenzen sind meistens Derivate der oben genannten Lizenzvereinbarungen. Sie folgen dem Geist der GPL mit der Ausnahme, dass sie den Lizenzgebern andere Bedingungen als allen anderen Nutzern gewähren.

Die folgende Tabelle stellt die individuellen Eigenschaften der verschiedenen Lizenzen dar.

Lizenz	Kann mit kommerzieller Software verwendet werden	Eigene Veränderungen müssen wieder frei sein	Kann unter anderen Bedingungen veröffentlicht werden	Enthält besondere Rechte für den Lizenzinhaber
GPL	Nein	Ja	Nein	Nein
LGPL	Ja	Ja	Nein	Nein
BSD	Ja	Nein	Nein	Nein
NPL	Ja	Nein	Nein	Ja
Public Domain	Ja	Nein	Ja	Nein

Tabelle 7: Darstellung der Eigenschaften der verschiedenen Lizenzen Quelle: o.V. 1999, S. 16

6.4.3 Tabellarische Übersicht der Open Source Lizenzen

Es gibt inzwischen eine recht große Anzahl an Lizenzen, die für sich in Anspruch nehmen, Open Source zu sein. Nicht alle davon entsprechen der Open Source Definition, noch weniger sind kompatibel zur GNU General Public License. Es gibt mehrere Versuche, eine Übersicht über die Lizenzen zu schaffen. Der folgende Überblick richtet sich weitgehend nach dem der FSF (Free Software Foundation).

Lizenz	**Copy-left**	**GPL komp.**	**frei**	**OSS (nach OSD)**	**Besonderheiten**

[30] vgl. o.V. 2003c

GPL	X	X	X	X	Siehe Abschnitt über GPL.
LGPL	-	X	X	X	Diese Lizenz ist im Gegensatz zur GPL nicht Copyleft, da sie ein Verlinken mit unfreien Modulen gestattet.
X11-Lizenz	-	X	X	X	Eine sehr einfache Lizenz, benutzt für XFree86.
modifizierte BSD-Lizenz	-	X	X	X	Eine sehr einfache Lizenz, die Advertising-Klausel der originalen BSD-Lizenz wurde entfernt.
Perl-Lizenz	-	X	X	X	Dies ist die Disjunktion von GPL und Artistic-License, nicht Copyleft, da Artistic nicht Copyleft ist, benutzt fast nur für Perl-Pakete.
Netscape Javascript Lizenz	-	X	X	X	Dies ist die Disjunktion von GPL und NPL, nicht Copyleft, da NPL nicht Copyleft ist.
originale BSD-Lizenz	-	-	X	X	Eine sehr einfache Lizenz, inkompatibel zur GPL durch die "obnoxious BSD advertising clause".
Apache-Lizenz	-	-	X	X	Eine sehr einfache Lizenz, inkompatibel zur GPL (sowohl Version 1.0 als auch Version 1.1).
Python-Lizenz	-	-	X	X	Lizenz für Python 1.6b1 und später (ab Version 2.0 gilt allerdings wiederum eine andere Lizenz). Inkompatibel zur GPL da die Gesetze von Virginia (USA) dafür gelten.
LaTeX Project Public License	-	-	X	X	Unvollständige Lizenz, inkompatibel zur GPL, jede veränderte Datei muss einen neuen Dateinamen bekommen, merkwürdige Definition von "Distribution".
MPL	-	-	X	X	Mozilla Public License, inkompatibel zur GPL.
NPL	-	-	X	X	Netscape Public License, Netscape behält sich besondere Rechte an dem Code vor (Möglichkeit, diesen in proprietären Programmen zu verwenden), inkompatibel zur GPL.
Sun Public License	-	-	X	X	Ähnlich zur MPL, nicht zu Verwechseln mit Sun Community Source License, inkompatibel zur

					GPL.
Qt Public License	-	-	X	X	unpraktisch, da modifizierte Sourcen nur als Patches verbreitet werden dürfen, wurde für die Qt-Library verwendet (steht jetzt unter GPL), inkompatibel zur GPL.
PHP Lizenz	-	-	X	X	PHP 3 dual-licensed unter PHP-Lizenz und GPL, PHP 4 nur noch unter PHP-Lizenz, Source-Split droht, da Version 3 und 4 parallel weiterentwickelt werden, inkompatibel zur GPL.
Artistic License	-	-	-	X	Ungenaue und vage Formulierungen.
APSL	-	-	-	X	Apple Public Source License, nicht frei, da Apple jederzeit die Erlaubnis zur Verbreitung der Software zurückziehen kann. Ab der Version 1.1 eine Open Source Lizenz nach der OSD. In der Version 1.2 erfolgte eine weitere Anpassung an die OSD, jedoch weiterhin nicht ausreichend für die FSF.
Sun Community Source License	-	-	-	X	Keine freie Lizenz, da man veränderte Versionen der Software nicht veröffentlichen darf.
Plan 9 Lizenz	-	-	-	-	Keine freie Lizenz, da man veränderte Versionen der Software nicht für sich behalten darf. Diese Lizenz ist auch von der OSI nicht als Open Source anerkannt.
Open Public License	-	-	-	X	Keine freie Lizenz, da man veränderte und veröffentlichte Versionen der Software immer an einen "initial developer" senden muss.

Tabelle 8: Vergleich der verschiedenen Open Source Lizenzen Quelle: Weber/Suhl 2001

6.4.4 Empfehlungen

Bevor nun konkrete Empfehlungen für die Wahl eines Lizenzmodells gegeben werden, wird zunächst ein Negativ-Beispiel aus der Praxis vorgestellt, aus dem dann einzelne Empfehlungen abgeleitet werden können.

Beispiel Mozilla[31] [32]**:**

Das wohl bekannteste Negativbeispiel für schlechte Lizenzierung eines OS-Projektes ist die Firma Netscape. Wie bereits erfahren, hat das Unternehmen den Quellcode seines Browsers veröffentlicht in der Hoffnung verlorene Marktanteile zurückgewinnen und unter dem Namen „Mozilla" weiterentwickeln zu können. Dazu hat Netscape einige Teile des Quellcodes unter die „Mozilla Public License" (MPL), andere hingegen unter die „Netscape Public License" (NPL) gestellt. Diese „Netscape Public License" beansprucht für Netscape das Recht, gegebenenfalls künftige Versionen des Quellcodes wieder unter eine proprietäre Lizenz zu stellen. Dies vertrug sich nicht mit dem Kodex in der Open Source Community. Dieser besagt, dass niemand besondere Eigentumsrechte für sich beanspruchen darf, die für andere Teilnehmer in der Open Source Community nicht gelten. Richard Stallmann, ein Vordenker der Open Source Bewegung, kritisierte die Philosophie der NPL vehement und empfahl, diese Lizenz nicht zu benutzen. Auch Eric Raymond warnte den damaligen CEO von Netscape Jim Barksdale vor solchen exklusiven Eigentumsrechten. Raymond wies Barksdale ausdrücklich darauf hin, dass die Zukunft von Mozilla vom Goodwill der Community abhängt. Er sagte voraus, dass der große Vertrauensvorschuss der Open Source Community durch exklusive Eigentumsrechte schnell zerstört sei. Die Community werde dann äußerst sensibel reagieren und keine Beiträge zum Gelingen von Mozilla leisten. Die Vorhersage von Raymond bewahrheitete sich. Durch die Sonderrechte von Netscape wurde die Norm des gleichen Nutzungsrechtes verletzt. Die Enttäuschung der Open Source Community war umso größer, weil die Erwartungen an dieses Open Source Projekt hoch gesteckt waren. In Folge arbeiteten größtenteils nur bezahlte Mitarbeiter am Open Source Projekt Mozilla, die Mehrheit der freiwilligen Programmierer verweigerte die Zusammenarbeit. Netscape hat aus seinen Fehlern gelernt. In der Zwischenzeit wurde die

[31] vgl. Osterloh/Rota/Kuster,2002
[32] vgl. Grassmuck 2002, S. 307 ff.

Lizenzstruktur von Mozilla so angepasst, dass eine Kompatibilität mit der „General Public License“ (GPL) gewährleistet ist. Im Juni 2002, vier Jahre nach Start des Mozillaprojektes, konnte eine stabile Version 1.0 veröffentlicht werden.

Dieses Beispiel zeigt, wie man es nicht machen sollte. Die Bedeutung der Community, von deren Know How der Erfolg eines solches Projektes abhängt, sollte nicht unterschätzt werden. Danach sollte auch die verwendete Lizenz gestaltet werden. Nach diesen Erfahrungen können wir nun einige Leitsätze auflisten, die es bei der Lizenzwahl zu beachten gibt und diese detaillierter erläutern:

Benutzen Sie eine der Open Source Definition entsprechende Lizenz![33]

Einen wichtigen Aspekt stellen die Anforderungen an eine OS-Lizenz dar. Damit eine Lizenz den Status einer OS-Lizenz erhält, muss diese bestimmte Anforderungen erfüllen. Diese Anforderungen hat die Open Source Initiative (OSI) in einer offiziellen Definition in der Form eines Kriterienkataloges zusammengefasst.[34]

Beachten Sie in jedem Fall die OS-Spielregeln!

Wie das vorangegangene Beispiel aufzeigt, ist es wichtig, dass Sie Ihre persönlichen Ziele nicht über die der Community stellen, falls beide Ziele miteinander konkurrieren. Möchten Sie zum Beispiel Ihre Software Open Source stellen, um das Bug Fixing voranzubringen, werden Sie nicht unbedingt von den Zielen der OS-Gemeinde abweichen, da diese ebenfalls an möglichst hochwertiger Software interessiert ist. Das Netscape-Beispiel zeigt aber auch, dass die Interessen und Ziele des initiierenden Unternehmens und der Gemeinschaft durchaus weit auseinander liegen können. In diesen Fällen ist es wichtig, die OS-Spielregeln einzuhalten. Wenn einmal der OS-Weg eingeschlagen ist, sollte von diesem nicht wieder abgewichen werden, da sonst eine gewisse Verärgerung bei den Community-Mitgliedern und sogar ein Boykott des Projektes zu befürchten wäre.

[33] vgl. Raymond/Lohmeier 2001
[34] vgl. OSD (2003)

Entwickeln Sie möglichst keine eigene Software-Lizenz![35]

Die weithin bekannten OSD-konformen Lizenzen haben eingeführte Traditionen der Auslegung. Programmierer (und, falls sie sich darum kümmern, auch User) wissen, was sie jeweils enthalten, und haben eine fundierte Meinung zu den Gefahren und Verlusten, die diese beinhalten. Benutzen Sie deshalb eine der auf der Seite der OSI zusammengetragenen Standardlizenzen, wann immer es Ihnen möglich ist.
Falls Sie doch Ihre eigene Lizenz schreiben müssen, so stellen Sie sicher, dass diese durch die OSI zertifiziert wird. Damit vermeiden Sie viele Diskussionen und Ballast.

Stellen Sie die Kompatibilität der Lizenz mit den Lizenzbedingungen dritter Software sicher!

Bei der Wahl einer Lizenz müssen eventuelle Kompatibilitätsfragen berücksichtigt werden (vgl. Schritt 3). Sind beispielsweise Programm-Bibliotheken oder auch komplette Programmbestandteile aus einer anderen freien Software übernommen, so sind die für das Urwerk eventuellen Lizenzbestimmungen auch für die Lizenz der eigenen Software relevant. Enthält z.B. die Lizenz des Hauptwerks eine Werbeklausel, so müssen auch alle Derivate diese Werbeklausel enthalten. Die Kompatibilitätsproblematik ist mitunter hochkomplex, da nicht immer klar definierbar ist, was Hauptwerk und was Derivat ist und welche Lizenzbedingungen letztlich dominieren.

Dasselbe gilt für den Fall der Verwendung von Applikationen zur Programmierung der Software. So sind eventuell die Lizenzbestimmungen des verwendeten Software-Toolkits bei der Auswahl bzw. Formulierung der eigenen Lizenz zu berücksichtigen.

[35] übernommen aus: vgl. Raymond/Lohmeier 2001

7 Schritt 6: Die Sofware in das richtige Format bringen

Der Aspekt der Softwareformatierung ist Inhalt dieses Prozessschrittes, der sich in der logischen Abfolge des Vorgehensmodells einer OS-Veröffentlichung an die Entscheidung für ein Lizenzmodell anschließt.

7.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Eine Open Source Software sowie ihr Quellcode sollten bestimmten formalen Ansprüchen genügen, damit diese für eventuelle Nutzer und Programmierer leicht zugänglich und verständlich ist. So steigert ein benutzerfreundliches Format der Software die Wahrscheinlichkeit einer breiten Nutzerschaft. Dies sind in diesem Kontext vor allem die späteren Weiterentwickler der involvierten Community sowie die Endanwender der Software. Die bisher entwickelte Software muss in diesem Schritt also soweit angepasst werden, dass sie nutzbar und möglichst anwenderfreundlich für die OS-Gemeinde ist.

7.2 Leitsätze und –fragen – Was muss beachtet werden?

Die hier beschriebene Anwenderfreundlichkeit betrifft in erster Linie die Software selbst sowie deren Quellcode. Ferner sind hier aber auch die Bereitstellung entsprechender Dokumentationen sowie die Distribution der Software angesprochen. Dementsprechend sind an die Anwenderfreundlichkeit in diesem Sinne einige Anforderungen zu setzen, die im Folgenden stichpunktartig dargelegt werden sollen.

- **Veröffentlichen Sie die Software unter einer neuen Versionsnummer oder einem neuen Namen!**

- **Stellen Sie für die Software Dokumentationen (z.B. Installationsdokumentationen etc.), How-To-Anleitungen o.ä. zur Verfügung!**

- **Bringen Sie den Source Code der Software in eine einfache und klare Struktur!**

Achten Sie auf eine richtige Namensgebung und Archivierungslogik!

Benutzen Sie Namen im GNU-Stil mit einem Namensstamm und einer major.minor.patch-Nummerierung!

Beachten Sie spezielle Konventionen!

Bemühen Sie sich um einen unverwechselbaren und leicht zu schreibenden Namen!

Benutzen Sie entweder eine gängige, in der Community gebräuchliche Programmiersprache (offene Standards wie PHP o.ä.) oder eine portable Skriptsprache!

- **Machen Sie Ihre Distribution anwenderfreundlich!**

Komprimieren Sie Ihre Dateien und stellen Sie sicher, dass Archive immer in ein einzelnes neues Verzeichnis entpackt werden!

Fügen Sie eine README-Datei bei!

Beachten und befolgen Sie die Benennung für die Standard-Dateien!

Achten Sie auf eine akzeptable Downloadgröße!

7.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

In der Umsetzung dieses Prozessschrittes werden vor allem technische Kompetenzen berührt. Die beschriebenen Aufgaben sollten idealerweise von den ursprünglichen Programmierern der Software erfüllt werden, da bei ihnen das entsprechende Software-Know-how liegt. Das Bild der Punkteverteilung sieht daher wie folgt aus:

Technisch	●●●●●
Rechtlich	
Betriebswirtschaftlich	

Tabelle 9: Kompetenzverteilung für Schritt 6

7.4 Erläuterungen

7.4.1 Veröffentlichen Sie die Software unter einer eigenen Versionsnummer oder einen neuen Namen!

Dies betrifft vor allem Software, die vor der OS-Veröffentlichung den Status einer proprietären Software inne hatte und dessen Lizenz von zahlreichen Kunden gekauft wurde. In diesem Fall könnte es ohne eine neue Versionsnummer oder –namen schwierig werden, diesen Kunden gegenüber zu rechtfertigen, warum die Software plötzlich kostenlos zur Verfügung steht. Auch zu befürchtende rechtliche Konsequenzen (vgl. Abschnitt 3.4) könnten für eine eigene Versionsnummer oder einen neuen Namen sprechen. Ein prominentes Beispiel stellt der Internetbrowser Netscape dar, der unter dem Namen Mozilla zu einer Open Source Software wurde.

7.4.2 Stellen Sie für die Software Dokumentationen, How-To-Anleitungen o.ä. zur Verfügung!

Außerdem sollten zu der Software entsprechende Dokumentationen, How-To-Anleitungen etc. angeboten werden. Diese Beigaben machen den Umgang mit der Software erheblich einfacher. So können sich einfache Anwender bequem Tipps und Anleitungen, beispielsweise zur Installation, aneignen. Auch Entwicklern würde dadurch weitergeholfen. So könnte ihnen beispielsweise die Portierung und Nutzung des Source-Codes erleichtert werden.

7.4.3 Bringen Sie den Source Code der Software in eine einfache und klare Struktur!

Die nachfolgenden Aspekte wurden aus Raymond/Lohmeier 2001 entnommen, die einen Leitfaden zur Veröffentlichung speziell von Linux-Software entwickelt haben. Die nachfolgenden Aspekte wurden daher derart modifiziert, dass sie einer Allgemeingültigkeit entsprechen.

Achten Sie auf die richtige Namensgebung und Archivierungslogik:
So wie das Uploaden auf Software-Archive wie Sourceforge, Metalab, oder das PSA- und CAPN-Archiv immer mehr zunimmt, gibt es einen fortschreitenden Trend, die Archivierung ganz oder teilweise automatisch durch Programme erledigen zu lassen (anstatt allein von Menschen). Deshalb wird es immer wichtiger, dass die Namen der Projekte und der Dateinamen dieser Archive korrekten Vorschriften entsprechen, so dass sie von Computerprogrammen erkannt und verstanden werden können.

Benutzen Sie Namen im GNU-Stil mit einem Namensstamm und einer major.minor.patch-Nummerierung!
Es ist allgemein hilfreich, wenn die Dateinamen Ihres Projektes den GNU-Namenskonventionen entsprechen - der Namensstamm zu Beginn (ausschließlich kleine Buchstaben und Zahlen), gefolgt von einem Gedankenstrich, dieser gefolgt von einer Versionsnummer, danach Erweiterungen und sonstige Zusätze.
Nehmen wir mal an, Sie haben ein Projekt namens `foobar' in der Version 1, Release 2, Level 3. Falls Sie daraus nur ein einziges Archiv bilden (vermutlich mit dem Quellcode), so sehen Sie im Folgenden, wie die Namen aussehen sollten.

- *foobar-1.2.3.tar.gz* - Das ist das Archiv mit dem Quellcode.
- *foobar.lsm* – Das ist die LSM Datei (sofern Sie sich an Metalab wenden).

Wenn Sie zwischen Quellcode- und Binär-Archiven zu unterscheiden haben, oder gar zwischen mehreren Binär-Archiven, oder im Dateinamen unterschiedliche Arten der Installation ausdrücken wollen, dann benutzen Sie dazu eine Namenserweiterung *hinter* der Versionsnummer. Das sollte dann etwa so aussehen:

- *foobar-1.2.3.src.tar.gz* **-** das Archiv mit den Quellen
- *foobar-1.2.3.bin.tar.gz* **-** Binärdateien ohne Spezifizierung
- *foobar-1.2.3.bin.ELF.tar.gz* - Binärdateien im ELF-Format
- *foobar-1.2.3.bin.ELF.static.tar.gz* **-** Binärdateien im ELF-Format, statisch gelinkt
- *foobar-1.2.3.bin.SPARC.tar.gz* **-** Binärdateien für SPARC-Maschinen

Benutzen Sie keinesfalls Namen wie `foobar-ELF-1.2.3.tar.gz', denn Programme tun sich schwer, Typbezeichnungen (wie `ELF') vom Namensstamm zu unterscheiden.

Ein nützliches Schema für einen Namen umfasst die folgenden Teile in der angegebenen Reihenfolge:

1. Stammname des Projekts
2. Gedankenstrich
3. Versionsnummer
4. Punkt
5. "src" oder "bin" (optional)
6. Punkt oder Gedankenstrich (Punkt bevorzugt)
7. Binärtyp und sonstige Optionen (optional)
8. Archivierungs- und Datenkompressions-Erweiterungen im Namen

Beachten Sie spezielle Konventionen!

Einige Projekte und Entwickler-Gemeinschaften haben genau definierte Konventionen für Namen und Versionsnummern, die nicht notwendigerweise mit den hier ausgeführten Ratschlägen übereinstimmen. Beispielsweise liegen Module von Apache generell in der Namensform mod_foo vor und haben sowohl pro Modul eine eigene Versionsnummer als auch für die Version von Apache selbst, zu der sie gehören. Ähnlich haben Module von Perl-Programmen Versionsnummern, die als Gleitpunktzahlen betrachtet werden können (so werden sie eher 1.303 finden als 1.3.3), und die Distributionen werden üblicherweise Foo-Bar-1.303.tar.gz genannt (für Version 1.303 des Moduls Foo::Bar). Schauen Sie, ob es für bestimmte Communities und Entwickler spezielle Namens-Konventionen gibt, und halten Sie diese ein.

Bemühen Sie sich um einen unverwechselbaren und leicht zu schreibenden Namen!

Der Namensstamm am Anfang sollte bei allen Projektdateien gleich sein, und er sollte einfach zu lesen, zu schreiben und zu merken sein. Deshalb benutzen Sie bitte keine Unterstriche. Und verwenden Sie keine großen Buchstaben am Anfang oder auch im Namen ohne wirklich guten Grund - das bringt die dem Menschen vertraute Suchfolge des Auges durcheinander und sieht nach einem Marketing-Zwerg aus, der versucht, schlau zu sein. Es verwirrt die Leute, wenn zwei verschiedene Projekte denselben Namensstamm haben. Versuchen Sie deshalb Kollisionen vor Ihrer ersten

Veröffentlichung festzustellen. Ein geeigneter Ort, um dies zu überprüfen, ist der index file of Metalab[36].

Benutzen Sie entweder eine gängige, in der Community gebräuchliche Programmiersprache oder eine portable Skriptsprache:

Der Portabilität und Stabilität zuliebe sollten Sie entweder gängige Programmiersprache (offene Standards wie PHP o.ä.) oder eine Skriptsprache benutzen, die Portabilität garantiert, weil sie über genau eine Implementation auf einer anderen Plattform verfügt. Geeignete Skriptsprachen sind u. a. Python, Perl, Tcl und Emacs Lisp. Einfache, alte Shells sind nicht geeignet, da es zu viele verschiedene Implementierungen mit subtilen Haarspaltereien von ihnen gibt und die Shell-Umgebungen Ziel von Durcheinanderbringungen durch Benutzer-Gewohnheiten sind. Beispielsweise verspricht Java Portabilität, aber die für Linux verfügbaren Implementationen sind noch problematisch und wenig in Linux integriert. Java ist noch eine riskante Wahl, obwohl es beliebter ist als es seiner Reife entspricht.

7.4.4 Machen Sie Ihre Distribution anwenderfreundlich!

Die folgenden Richtlinien beschreiben Tipps für die Distribution der Software, also für den Fall, das Nutzer sie aus dem Internet herunterlädt, wiederherstellt und entpackt. Die folgenden Aspekte wurden zum Teil aus Raymond/Lohmeier 2001 entnommen.

Komprimieren Sie Ihre Dateien und stellen Sie sicher, dass die Archive immer in ein einzelnes neues Verzeichnis entpackt werden![37]

Da die zum Download bereitgestellten Dateien in der Regel groß sein werden, sollten Sie Ihre Dateien entsprechend mit gängigen Komprimierungsprogrammen (z.B. Winzip, Zip, Pkzip, Gzip, Tar) packen.

Ärgerlich für den Anwender ist, Archive zu bilden, die die darin enthaltenen Dateien und Verzeichnisse direkt ins aktuelle Verzeichnis entpacken. Dabei könnten bereits existierende Dateien überschrieben werden. Stellen Sie deshalb sicher, dass alle Dateien in ein gemeinsames Verzeichnis entpackt werden, das den Namen des

[36] http://metalab.unc.edu/pub/Linux
[37] entnommen aus: Raymond/Lohmeier 2001

Projekts trägt. Auf diese Weise werden sie in ein einziges oberstes Projekt-Verzeichnis entpackt, dass direkt im aktuellen Verzeichnis wurzelt.

Fügen Sie eine README-Datei bei![38]

Eine Datei mit dem Namen README oder READ.ME sollte vorhanden sein; sie ist eine Landkarte Ihrer Software-Distribution. Nach alter Sitte ist dies die erste Datei die unerschrockene Neugierige nach der Entpackung der Quellen lesen werden.

In der README-Datei werden sich als nützlich erweisen:

- eine kurze Projekt-Beschreibung
- ein Hinweis auf die Website des Projekts (sofern vorhanden)
- Hinweise auf die Entwicklungsumgebung des Programmierers und auf mögliche Portabilitätsprobleme
- ein Liste, die wichtige Dateien und Unterverzeichnisse beschreibt
- eine Installationsanleitung oder einen Hinweis auf eine Datei, die diese enthält (meist INSTALL)
- eine Liste der Unterstützer und Danksagungen oder ein Hinweis auf eine Datei, die diese enthält (meist CREDITS)
- aktuelle Projekt-Neuigkeiten oder ein Hinweis auf eine Datei, die solche enthält (gewöhnlich ist dies NEWS).

Beachten und befolgen Sie die Benennung für die Standard-Dateien!39

Bevor er noch nach der README-Datei schaut, wird der neugierige Anwender die Dateinamen im obersten Verzeichnis Ihrer entpackten Distribution gelesen haben. Diese Namen können bereits der Information dienen. Wenn Sie bestimmte eingebürgerte Namensgebungen einhalten, können Sie dem Erkundenden wertvolle Hinweise geben, wonach er als Nächstes Ausschau halten sollte.

Hier sind nun einige Dateinamen, die üblicherweise im höchsten Ordner zu finden sind, und deren Bedeutung. Natürlich muss nicht jede Distribution alle davon besitzen.

- **README oder READ** - die "Landkarte" der Distribution; als Erstes zu lesen
- **INSTALL** - Anweisungen zur Konfiguration und Installation
- **CREDITS** - Liste derer, die etwas zum Projekt beigetragen haben

[38] entnommen aus: Raymond/Lohmeier 2001
[39] entnommen aus: Raymond/Lohmeier 2001

- **NEWS** - laufende Projekt-Neuigkeiten
- **HISTORY** - Geschichte des Projekts
- **COPYING** - Lizenzbestimmungen für das Projekt (GNU Konvention)
- **LICENSE** - Lizenzbestimmungen für das Projekt
- **MANIFEST** - Verzeichnis der Dateien der Distribution
- **FAQ** - Liste mit häufig gestellten Fragen zum Projekt in einfachem Textformat
- **TAGS** - Datei mit `tags' für die Editoren vi oder Emacs

Beachten Sie die weit verbreitete Konvention, dass Dateinamen, die nur aus Großbuchstaben bestehen, keine Programmkomponenten sind, sondern von Menschen lesbare Metainformationen bezüglich der Software darstellen. Eine FAQ kann viel Ärger ersparen. Wenn bestimmte Fragen zum Projekt öfters auftauchen, dann packen Sie diese Fragen in die FAQ; halten Sie die Benutzer an, erst die FAQ zu lesen, bevor sie Fragen losschicken oder Fehler melden. Eine gut aufgebaute FAQ kann den Support, der auf den Projekt-Unterstützern lastet, vermindern.

Eine HISTORY- oder NEWS-Datei mit einer zeitlichen Übersicht über alle Versionen ist nützlich. Unter anderem kann es Kniffe früher einführen, falls Sie jemals mit einem Patentverletzungs-Verfahren in Berührung kommen sollten.

Achten Sie auf eine akzeptable Downloadgröße!

Wenn Sie Ihre Software ausschließlich über das Internet vertreiben, sollten Sie darauf achten, dass Ihre Distribution in ihrer Größe nicht den Rahmen sprengt. Das könnte potenzielle Nutzer, weniger die interessierten Entwickler, frühzeitig abschrecken. Und falls Sie Ihr Geschäftsmodell auf eine möglichst breite Anwendergruppe ausgerichtet haben, sollten Sie diesen Aspekt nicht einfach übergehen.

8 Schritt 7: Entscheidungen bezüglich der Distribution der OSS und der Betreuung der Community treffen

Mit der Distribution der Software ist hier die eigentliche physische Verteilung der Software gemeint. Zur Betreuung der Community zählen unter anderen Aktivitäten wie das Einsammeln von Verbesserungsvorschlägen der Mitglieder, die Moderation von Foren, das Bereitstellen von Informations- und Kommunikationskanälen etc. Eine bedeutende Entscheidung für das vorliegende Vorgehensmodell muss innerhalb dieses Schrittes nun dahingehend getroffen werden, ob diese Aufgaben in Eigen- oder Fremdleistung erbracht werden sollen.

Die Distribution und die Community-Betreuung werden in der Praxis sehr häufig von den gleichen Aufgabenträger übernommen, obwohl diese Aufgaben technisch gesehen natürlich auch getrennt voneinander durchführbar wären. Für dieses Vorgehensmodell soll der gängigen Praxis entsprochen werden und auf die Möglichkeit einer getrennten Betrachtung verzichtet werden. Entsprechend bedingt ein Eigenvertrieb aus der hier betrachteten Perspektive eine spätere Community-Betreuung.

Vom logischen Ablauf des Vorgehensmodells kann dieser Schritt parallel zum vorherigen Schritt verlaufen (Schritt 6). Es ergeben sich keinerlei Abhängigkeiten dieser beiden Schritte (vgl. Anhang B). Nach Bearbeitung dieses Schrittes schließt sich unabhängig von der oben zu fällenden Entscheidung der Schritt der Abschlusskontrolle (Schritt 8) an. Ob darüber hinaus der Parallelzweig im Vorgehensmodell (vgl. Anhang B) bestehend aus Schritt 10 und 11 eingeschlagen wird, hängt davon ab, ob in der obigen Entscheidungssituation die Möglichkeit der Eigendistribution und der damit verbundenen Community-Betreuung gewählt wurde. Ist dies der Fall, so müssen Schritt 10 und 11 jeweils parallel zu den Schritten 8 und 9 bearbeitet werden. Bei dieser jeweiligen Parallelbearbeitung ergeben sich keine logischen Abhängigkeiten zwischen den Schritten 8 und 10 sowie 9 und 11.

8.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Ziel dieses Prozessschrittes ist es, eine Entscheidung über eine Eigen- oder Fremddistribution der Software sowie eine langfristige Betreuung der Community zu

fällen. Als Informationsgrundlage für diese Entscheidung dienen dabei im Wesentlichen die Ergebnisse aus Schritt eins, in dem geklärt wurde, auf welche finanziellen, personellen und sachlichen Ressourcen zurück gegriffen werden kann. Zudem stellt die Zielsetzung (Schritt 3) eine Entscheidungsgrundlage dar. So ist sicherlich eine Distribution in Eigenregie sinnvoller, wenn man mit der OS-Strategie ein Zusatzgeschäft betreiben will. Im Gegensatz dazu ist eine Fremddistribution für eine bloße Veröffentlichung der Software sinnvoller.

8.2 Leitsätze und –fragen – Was muss beachtet werden?

Zunächst sollte grundsätzlich geklärt werden, ob die Distribution der Software sowie die Community-Betreuung nicht von anderen übernommen werden kann. Diese Entscheidung hängt im Wesentlichen von den gesteckten Zielen ab. So ist es für das Ziel einer bloßen Weitergabe der Software sicherlich sinnvoll, die Distribution in andere Hände zu übergeben und somit eigene Ressourcen zu schonen.

Oftmals ist es allerdings aus strategischen Gründen geeigneter, die Distribution und Community-Betreuung selbst zu übernehmen. Dies kann beispielsweise sinnvoll sein, wenn das erklärte Ziel ist, von der Weiterentwicklung und dem Bug-Fixing durch die Community profitieren zu wollen (vgl. Zielsetzung Kapitel 4). Die Kontrolle zu behalten, ist auch sinnvoll, wenn das Ziel die Vermarktung von Zusatzleistungen ist. In diesen Fällen sollte die OSS veröffentlichende Institution die Kontrolle über den weiteren Entwicklungsverlauf der Software im Sinne einer moderierenden Instanz der Community-Bewegung beibehalten. Auf diese Weise kann weiterhin ein indirekter Einfluss im Sinne der Zielsetzung auf die zukünftige Entwicklung der Software ausgeübt werden. In diesem Fall sollten allerdings die dafür benötigten und nicht unerheblichen finanziellen, personellen und sachlichen Ressourcenaufwände einkalkuliert werden.

8.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Dieser Prozessschritt benötigt für seine Umsetzung sowohl technisches als auch betriebswirtschaftliches Know-how. Es werden auf technischer Seite konkret Kenntnisse im Umfeld des Internets und des CVS benötigt. Die betriebswirtschaftlichen Kompetenzen sollten konkret aus dem Marketingbereich

sowie des strategischen Managements kommen. Eine Punkteverteilung ergibt sich deshalb zu Gunsten der technischen und betriebswirtschaftlichen Kompetenzen:

Technisch	●●◖
Rechtlich	
Betriebswirtschaftlich	●●◖

Tabelle 10: Kompetenzverteilung für Schritt 7

8.4 Erläuterungen

Wie dargelegt, ist eine Entscheidung über die Distribution der Software sowie die Betreuung der Community zu fällen. Der wesentliche Vorteil der eigenverantwortlichen Übernahme dieser Aufgaben ist die Beibehaltung der Kontrolle über die Software. Dies beinhaltet die Kontrolle über die Software selbst sowie deren Source Code. Auf diese Weise kann auch in Zukunft indirekt ein Einfluss auf die Fortentwicklung der Software ausgeübt werden. Wie gezeigt, kann dies für einige Zielsetzungen der OS-Veröffentlichung strategisch günstig sein.

Für den Fall des Eigenvertriebs der Software sind in dem hier behandelten Vorgehensmodell zwei wesentliche Schritte (Nr. 10 und 11) in die Planungen der OS-Veröffentlichung einzubeziehen. Dies sind zum einen die Schaffung der technischen Voraussetzungen (Schritt 10), um den Vertrieb der Software sowie die Gründung einer Community gewährleisten zu können. Dazu gehört unter anderem die Schaffung eines Podiums im Internet, das Diskussionsforen, Downloadmöglichkeiten der Software, Newsveröffentlichungen etc. bietet. Außerdem gehört die Gründung und Initiierung einer Community (Schritt 11) zu den zusätzlichen Aufgaben.

Die hier angesprochenen Schritte 10 und 11 gehören zu den einmaligen Tätigkeiten. Zum anderen ist die bereits angesprochene Betreuung der Community eine dauerhafte Aufgabe, für die ein nicht zu unterschätzender Aufwand betrieben werden muss. Entsprechend der hier angedeuteten und im weiteren Verlauf detaillierter beschriebenen Aufwände für den Fall eines Eigenvertriebes muss ein adäquater Ressourceneinsatz eingeplant werden, der sowohl von personeller als auch finanzieller und sachlicher Art ist. Bei der Entscheidung für einen Eigenvertrieb sollte dieser Aufwand einkalkuliert werden.

Dieser Aufwand ergibt sich natürlich nicht, wenn auch ein Fremdvertrieb möglich ist. Dieser bietet sich beispielsweise für öffentliche Einrichtungen an, die die aus Forschungsarbeiten resultierende Software lediglich veröffentlichen möchten und deren strategischen Ziele und Absichten nicht in dem Maße oder gar nicht wie im obigen Fall von dem zukünftigen Entwicklungsverlauf der Software abhängen. Wie beschrieben, ist mit der Abgabe der Distribution allerdings auch ein Verlust einer gewissen Kontrolle über das weitere Schicksal der Software verbunden, der nicht ohne weiteres wett- gemacht werden kann. Allerdings spielt der Kontrollverlust für einige Zielsetzungen oftmals sowieso eine eher untergeordnete bzw. vernachlässigbare Rolle.

Für den Fremdvertrieb sowie die Abgabe der Community-Betreuung eigenen sich Open Source-Portale im Internet. Open Source Portale bringen Anbieter und Nachfrager von OSS wie auf einen Marktplatz zusammen. „Sie dienen als Vermittler zwischen Angebot und Nachfrage und versuchen zwischen Entwicklern, Anwendern, Distributoren und kommerziellen Softwareanbietern zu vermitteln. Außerdem fungieren die Mediatoren auch als Entwicklungsplattform, die für die Entwicklung von Open Source Software zur Verfügung stehen.“[40] Zu den bekanntesten OS-Portalen gehören beispielsweise SourceForge und BerliOS.

[40]vgl. Hang/Hohensohn 2003, S. 60

9 Schritt 8: Abschlusskontrolle vor der Softwarefreigabe

9.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Durch diesen Arbeitsschritt soll vor der Freigabe der Software noch einmal sichergestellt werden, dass die Bearbeitung der vorangegangenen Arbeitsschritte gemäß der Ziel- und Strategievorgaben (Schritte 3 und 4) erfolgten. Zudem sollte eine Abstimmung der Teilergebnisse spätestens in diesem Arbeitsschritt erfolgen. Diese Aspekte sind deshalb so wichtig, da mit dem nächsten Schritt der Softwarefreigabe Nachbesserungen oder Behebungen eventuell gemachter Fehler kaum möglich sind. Dieser Prozessschritt stellt also die letzte Möglichkeit dar, die operative Umsetzung der Ziel- und Strategievorgaben zu kontrollieren.

9.2 Leitsätze und –fragen – Was muss beachtet werden?

Für diese Kontrolle der operativen Arbeitsschritte im Vorgehensmodell sollte ein systematischer Reviewprozess erfolgen. Dabei eventuell entdeckte Schwachstellen und Fehler sollten umgehend an und von entsprechender Stelle behoben werden.

Ein Reviewprozess sollte nach Möglichkeit so systematisiert werden, dass jeder von dem Prozessverlauf berührte Kompetenzbereich ein eigenes Review seiner durchgeführten Aufgaben vornimmt. Somit ergeben sich Reviews für den technischen, rechtlichen und betriebswirtschaftlichen Bereich.

Nachfolgend werden für jeden Kompetenzbereich einige exemplarische Aspekte aufgezählt, die im Kontrollvorgang geprüft werden sollten. Je nach Ausrichtung des Projektes werden zudem zu kontrollierende Detailaspekte auftreten, die auf Grund der hier angestrebten Allgemeingültigkeit und Prägnanz aber nicht behandelt werden sollen.

Rechtliche Aspekte:

Sind wirklich alle Rechtsverhältnisse an dieser Software geklärt? / Gibt es wirklich niemanden, der nach der Veröffentlichung Ansprüche geltend machen kann?

Ist für die gewählte Ziel- und Strategieplanung die richtige Lizenz gewählt? / Passt die Lizenz zur konkretisierten Strategie?

Sind alle formalen Aspekte bei der Lizenzierung der Software beachtet worden?

Technische Aspekte:

Ist die Software so umgestaltet bzw. erstellt worden, dass sie veröffentlicht werden kann?

Gibt es genügend Dokumentation dazu?

Ist die Hardwareumgebung richtig konfiguriert?

Für den Fall des Eigenvertriebs der Software: sind alle notwendigen Kommunikations-, Informations- sowie Absatzkanäle eingerichtet?

Betriebswirtschaftliche Aspekte:

Ist das Geschäftsmodell ausreichend an den vorgegebenen Zielen ausgerichtet?

Sind die Zuständigkeiten für den weiteren Projektverlauf geklärt?

Sind die eventuellen Angebote eines Zusatzgeschäftes soweit ausgereift, dass sie nach der OS-Veröffentlichung auf dem Markt angeboten werden können?

Stehen alle benötigten Ressourcen bereit und passen diese in das geplante Budget?

Neben der Kontrolle dieser fachspezifischen Fragestellungen gehört es zudem zur Aufgabe der Projektleitung, spätestens im Rahmen dieses Arbeitsschrittes sicherzustellen, dass die Ergebnisse sämtlicher Teilschritte dieses Vorgehensplans zueinander kompatibel sind und miteinander harmonieren. Dies ist für die Erzielung eines in sich schlüssigen Gesamtergebnisses von besonderer Bedeutung.

9.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Die Erfüllung der Aufgaben erfordert alle Kompetenzen. So obliegt es der rechtlichen Kompetenz alle rechtlichen Aspekte zu kontrollieren. Entsprechendes gilt für die technischen und betriebswirtschaftlichen Aufgabenträger. Bei der Punkteverteilung soll der betriebswirtschaftlichen Kompetenz allerdings ein größeres Gewicht zukommen, da der Kontrollaufwand für diesen Kompetenzbereich entsprechend höher ausfallen dürfte.

Technisch	•◂
Rechtlich	•◂
Betriebswirtschaftlich	••

Tabelle 11: Kompetenzverteilung für Schritt 8

10 Schritt 9: Freigabe der Software (inklusive des Source Code)

Ist der Kontrollvorgang abgeschlossen und kann davon ausgegangen werden, dass eventuelle Schwachpunkte festgestellt und beseitigt wurden, kann die Software freigegeben werden. Dies bedeutet einerseits, dass im Falle eines Fremdvertriebes die Software einschließlich des Source Code an ein Intermediär (OS-Portal) übergeben werden kann. In diesem Fall sind diesem dann entsprechende Rechte an der Software einzuräumen. Dazu gehört beispielsweise der schon angesprochene Verzicht des Urheberrechtes (vgl. Schritt 2). Der Prozess der OS-Veröffentlichung wäre in diesem Fall beendet. Das weitere Schicksal der Software ist somit in fremde Hände gelegt und nicht mehr beeinflussbar.

Hat Schritt 7 allerdings ergeben, dass die Distribution und die Community-Betreuung in Eigenregie erfolgen soll, so ist nach der hier beschriebenen Freigabe der Software der Start der eigenverantwortlichen Distribution abgesegnet. In diesem Fall kann die Software einschließlich des Source Codes veröffentlicht werden. Konkret heißt dies, dass die Software auf der Internetseite zum Download angeboten werden sollte. Die entsprechenden absatztechnischen Voraussetzungen (Schritt 10) sollten bis dahin gegeben sein. Parallel zur Freigabe der Software kann dann mit der Gründung einer Community (Schritt 11) begonnen werden.

Da dieser Schritt auf Grund der getroffenen Vorbereitungen keinen besonderen Aufwand mehr erfordert, ist dieser Aspekt weniger als eigenständiger Arbeitsblock zu verstehen, sondern eher als Milestone. Nach Erreichen dieses Milestones und mit Beendigung der einmaligen Vorarbeiten tritt das Projekt in eine Phase, in der die Kräfte darauf konzentriert werden sollten, das langfristige Überleben des Projektes zu sichern.

11 Schritt 10: Absatz- und projekttechnische Voraussetzungen schaffen

11.1 Ziele - Was soll durch diesen Schritt erreicht werden?

Ist die Entscheidung in Schritt 7 zu Gunsten eines Eigenvertriebes und der dazugehörigen Community-Betreuung gefallen, so sind in diesem Schritt die entsprechenden Voraussetzungen dafür zu schaffen. Ziel ist es, einen auf Dauer angelegten Absatzkanal für die Software, Kommunikations- und Informationskanäle für die Community-Betreuung und die Mitglieder untereinander sowie Entwicklungsumgebungen für die Zusammenarbeit der Community-Mitglieder zu schaffen.

Ein wichtiges Medium dafür stellt das Internet dar. Zwar sind speziell für den Absatz der Software auch andere Kanäle denkbar (etwa Verteilung der OSS auf CD-Rom in Fachzeitschriften), da aber in der Praxis das Internet den am häufigsten gewählte Absatzkanal darstellt, soll hier auf eine Betrachtung anderer Absatzkanäle verzichtet werden.

Für die Betreuung der Community stellt das Internet einen geradezu idealen Kommunikationskanal dar. Auch in diesem Zusammenhang soll hier ausschließlich das Internet als entsprechendes Medium betrachtet werden.

Zusammenfassend ist es also das Ziel dieses Schrittes, einen entsprechenden Internetauftritt zu gestalten, der die hier erwähnten Aspekte der Kommunikation, Information, Distribution und Kooperation vereinigt.

11.2 Leitsätze und –fragen – Was muss beachtet werden?

Für die Kommunikation und Information der Community-Mitglieder über das Internet sollte zunächst eine eigene Homepage eingerichtet werden. Beispielsweise stellen Mailinglisten, Diskussionsforen und Newsgroups wichtige Elemente in diesem Zusammenhang dar. Die Implementierung entsprechende Entwicklertools in die Internetseiten erleichtern zudem die Zusammenarbeit der Projektmitglieder. Bezüglich der Distribution sollte entsprechend ein Downloadbereich eingerichtet

werden. In diesem Bereich sollte einerseits der Source Code zur Verfügung stehen und andererseits natürlich die Software selber.

11.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Dieser Prozessschritt bedarf vor allem technischer Kompetenzen. Entsprechend sollten hier IT-Fachkräfte mit Erfahrungen im Internetbereich eingesetzt werden. So ergibt sich eine Punkteverteilung gemäß der folgenden Abbildung:

Technisch	●●●●●
Rechtlich	
Betriebswirtschaftlich	

Tabelle 12: Kompetenzverteilung für Schritt 10

11.4 Erläuterungen

Wie bereits dargelegt, stellt das Internet für die Eigendistribution der Software den am häufigsten benutzten sowie geradezu idealen Absatzkanal dar. Durch den Wegfall der Speicherung auf CD-Rom können durch diesen Absatzweg enorme Kosten eingespart werden. So sollte die Erstellung einer Internetpräsenz unter anderem aus diesem Grund erfolgen. Entsprechend ist ein umfangreicher Download-Bereich bei der Gestaltung der Internetseiten einzurichten, von dem aus die Software und der dazugehörige Source Code von potenziellen Nutzern und Community-Mitgliedern heruntergeladen werden kann.

Der wichtigste Kommunikations- und Informationskanal von OS-Projekten ist das Internet. So werden alle Aktivitäten eines OS-Projektes über das Internet geplant, koordiniert, durchgeführt und kontrolliert. Auch dazu bedarf es in erster Linie einer projekteigenen Website[41].

Neben der Kommunikation ist aber auch die Information wichtig. So dient die Website auch der Präsentation des Projektes, insofern stellt sie sein Spiegelbild des Projektes dar. Dementsprechend kommt es auf die qualitativ hochwertige Gestaltung der Website an. Auch thematisch sollte die Website einige Standards erfüllen. So

[41] vgl. DBUS o. J.; Raymond/Lohmeier 2001

verfügen die Internetpräsenzen erfolgreicher OS-Projekte häufig über folgende Themenschwerpunkte[42]:

- Neuigkeiten, Ankündigungen, Berichte über das Projekt
- die "Charta", also die Grundsatzerklärung des Projekts (wozu es dient, wer die Zielgruppe ist, usw.)
- Links zum Downloaden der Projekt-Quellen
- Startseiten für Mailinglisten des Projekts
- eine FAQ (Frequently Asked Questions)-Liste
- die Projekt-Dokumentation in HTML-Format
- Links zu in Beziehung stehenden und/oder konkurrierenden Projekten

Bezüglich der Kommunikation sollten Mailinglisten eingerichtet werden[43]. Kommunikationskanal Nummer eins sind E-Mail-Verkehr und Mailing-Listen. Sie fungieren als elektronische Rundschreiben, die jeder enthält, der sich in entsprechende Listen eingetragen hat. Aufgeteilt nach Themen wie Design, Dokumentation, Lizenzfragen, Plattform und speziellen Softwarekomponenten werden Neuigkeiten ausgetauscht, geplant, diskutiert und abgestimmt.

Auch die Einrichtung von Newsgroups ist wichtig[44]. Newsgroups funktionieren wie Mailinglisten auf der technologischen Basis von E-Mail-Systemen. Sie werden wie Mailinglisten für die Einwege-Kommunikation genutzt und können von den Community-Mitgliedern abonniert werden. Während bei Mailing-Listen jedoch die Anzahl der Empfänger wegen der oft hochspeziellen Themen meist relativ gering gehalten wird, ist sie bei Newsgroups unbeschränkt. Statt zur spezifischen Erarbeitung von Details zur Softwareentwicklung werden sie in der Praxis eher für allgemein gehaltene Informationen und Ankündigungen genutzt.

Häufig erweisen sich auch Diskussionsforen als nützlich[45]. Praktikabel hierbei ist die Einführung spezieller Kategorien, so dass Benutzer nach Interessenschwerpunkten vorselektieren können. Ein Problem solcher Diskussionsforen könnte allerdings eine

[42] in Anlehnung an Raymond/Lohmeier 2001
[43] vgl. DBUS o. J.; Raymond/Lohmeier 2001
[44] vgl. DBUS o. J.; Raymond/Lohmeier 2001
[45] vgl. Behlendorf 1999

hohe Nutzerzahl darstellen. So könnten die entsprechenden Server schnell überlastet sein. Also empfiehlt sich die Einführung von Diskussionsforen eher für OS-Projekte mit einer entsprechend überschaubaren Anzahl an Nutzern.

Unerlässlich für jedes Projekt ist zudem die Einrichtung spezieller Entwicklertools, über die die Zusammenarbeit der Community-Mitglieder vereinfacht wird[46]. Die durch unzählige Programmbeiträge lebenswichtig gewordene Organisation des Quellcodes übernimmt beispielsweise eine Konfigurations- und Versionierungssoftware. Im Open-Source-Sektor hat sich das Concurrent Versioning System (CVS) durchgesetzt, das für zahlreiche OS-Plattformen existiert. In Baumform werden aber nicht nur Quelltext, sondern auch Objektcode, Binärdateien und Dokumentation in einer Datenbank verwaltet. Oft wird auch eine Fehler-Datenbank gepflegt, die von Mitgliedern über ein Web-Interface angesprochen werden kann[47]. Das Bug-Tracking und -Fixing wird so deutlich vereinfacht. So wird beispielsweise im Apache Projekt das standardisierte Tool GNATS benutzt.

Die Datenbank eines CVS lässt sich am besten als ein Repository beschreiben, das alle Entwicklungsinformationen, die beim Aufbau eines Programms anfallen, enthält. Dabei handelt es sich gleichermaßen um Methoden und Daten sowie um Ausgangs-, Zwischen- und Endergebnisse aller Art, die darin abgelegt werden. Auch die beteiligten Personen und Werkzeuge können über das Repository koordiniert werden.

Features von CVS:

- Festhalten des Ablaufes aller Veränderungen
- Bietet Unterstützung bei der Herstellungskontrolle und Kontrolle der Änderungen
- Unterstützt den Zugang in die Verzeichnisse von remote hostes
- Kontrolliert und vermeidet Doppelarbeiten, so dass ein gleichzeitiges Arbeiten mehrer Programmierer an demselben Source Codes zur selben Zeit möglich ist

[46] vgl. DBUS o. J.
[47] vgl. Behlendorf 1999

12 Schritt 11: Gründung einer Community

Vom logischen Ablauf des Vorgehensmodells schließt sich dieser Schritt an die Schaffung der absatz- und projekttechnischen Voraussetzungen (Schritt 10) an. Mit Beginn dieses Schrittes sollte allerdings parallel die Freigabe der Software (Schritt 9) erfolgen, da ohne die Software die Gründung einer Community keinen Sinn ergeben würde. Nach Abarbeitung der Schritte 10 und 11 bzw. 8 und 9 sollte letztlich aus dem Prozess der OS-Veröffentlichung als Ergebnis ein auf Dauer angelegtes OS-Projekt hervorgegangen sein.

12.1 Ziele – Was soll durch diesen Schritt erreicht werden?

In diesem Schritt geht es im weitesten Sinne darum, eine Community um die Software herum zu planen sowie aufzubauen und zu etablieren. Die Community stellt gewissermaßen den Motor für ein OS-Projekt dar, ohne den die OS-Idee zum scheitern verurteilt wäre.

Ziel dieses Schrittes soll es daher sein, auf der Basis der Informationen der vorangegangenen Schritte zunächst die Gründung einer Community detailgetreu zu planen. Desweiteren gehört es zum Ziel dieses Prozessschrittes, die Gründung der Community zu initiieren. Zur Bewältigung dieser Aufgaben sind dabei sowohl soziale als auch personelle Fragen zu klären. Auf diese Aspekte soll in den jeweils folgenden Abschnitten dieses Kapitels näher eingegangen werden.

12.2 Leitsätze und –fragen – Was muss beachtet werden?

12.2.1 Soziale Aspekte:

Gestalten Sie eine Organisationsstruktur für die Community!

Eine OS-Community ist üblicherweise unstrukturiert und besteht aus mehreren tausend Mitgliedern. Um nun aber einen Nutzen aus der Community ziehen zu können, bietet es sich an, bei der Gründung einer Community entsprechende Strukturen zu etablieren und die Mitgliederzahl überschaubar zu halten. Ein entsprechendes organisatorisches Management bei der Gründung einer Community sollte also vorausgesetzt sein. Dabei ist aber darauf zu achten, dass die Regeln und Grundsätze der OS-Idee nicht verletzt werden.

Versuchen Sie eine eigene Makrokultur zu etablieren!

Die Förderung und Etablierung einer Kultur für die Community wirkt neben einem Zusammenhalt der Mitglieder vor allem in zwei Richtungen. Zum einen kann eine Kultur durch die von den Mitgliedern gemeinsam geteilten Werte, Einstellungen und Normen sehr interdependente und komplexe Aktivitäten, wie zum Beispiel OS-Projekte, koordinieren und fördern. Aus demselben Grund wirkt sie deshalb auch als Kontroll- bzw. Sanktionsmechanismus. Der Aufbau einer Kultur kann in diesem Kontext also als ein wichtiges und nützliches Instrument angesehen werden und sollte daher nicht vernachlässigt werden.

Schaffen Sie Motivationen für die Mitglieder!

Einen wichtigen Faktor zur Beeinflussung der Motivation stellt die Reputation der Community-Mitglieder. So sollten von Seiten der Projektführung Instrumente entwickelt werden, die eine Reputation der Mitglieder fördern können. Dazu gehören zum Beispiel Auszeichnungen für besondere Programmierleistungen, das Ausstellen von Referenzen oder Belobigungen einzelner Mitglieder für Teilerfolge in öffentlichen Medien.

12.2.2 Personelle Aspekte

Schaffen Sie Schnittstellen!

Nach der Initiierung der Gründung sollte sich die Community weitestgehend selbst am Leben erhalten. Das heißt aber nicht, dass von dort an der Kontakt zur Community gänzlich abgebrochen werden sollte. Im Gegenteil: Um einen Nutzen im Sinne der gesteckten Ziele aus der Community ziehen zu können, muss dieser Kontakt aufrechterhalten und gepflegt werden. Ziel sollte es sein, das Projekt in einer Moderatorenfunktion zu betreuen. Dafür sollten im Rahmen dieses Prozessschrittes verschiedene Schnittstellen eingerichtet werden. Derartige Schnittstellen sollten durch die Einrichtung der folgenden Rollen abgedeckt werden.

- **Organisator, Antreiber, Stratege:** *Impulsgeber für das OS-Projekt, der das Projekt gemäß den Zielen in die richtigen Bahnen lenkt*
- **Infrastrukturbetreuer:** *technischer Betreuer für die Kommunikationskanäle*

- **Code Käpt'n:** *Verantwortlicher für den Source Code*
- **Bug-Verwalter:** *Verantwortlicher für die Bugs*
- **Redakteur:** *Verantwortlicher für die Softwaredokumentation sowie die Pflege der Projekt-Website*

12.3 Prozessbeteiligte – Wer ist für diesen Schritt zuständig?

Dieser Prozessschritt erfordert betriebswirtschaftliche Kompetenzen. Mögliche Kompetenzträger sollten dabei sowohl aus den Teildisziplinen Personalwirtschaft und Organisationstheorie stammen. Es ergibt sich dementsprechend folgendes Bild:

Technisch	
Rechtlich	
Betriebswirtschaftlich	●●●●●

Tabelle 13: Kompetenzverteilung für Schritt 11

12.4 Erläuterungen

12.4.1 Soziale Aspekte

Die Verbindung der Organisation, die ihre Software OS stellen möchte, mit der Community stellt eine besondere Beziehung dar. Das liegt daran, dass die Mitglieder der Community einen gänzlich anderen Status genießen, als es die Mitwirkenden von Projekten, Unternehmen, Forschungseinrichtungen etc. tun. Während letztere für ihre Aktivitäten bezahlt werden, geschieht das Engagement der Community-Mitglieder auf freiwilliger Basis. Entsprechend wird im ersten Fall auch von Humankapital, im zweiten Fall aber von sozialem Kapital gesprochen[48]. Analog lassen sich die Mitglieder der Community auch nicht in hierarchische Strukturen einordnen, und die Beziehung zwischen diesen Parteien kann auch nicht als eine Marktbeziehung bezeichnet werden, sondern eher als eine soziale Beziehung. Sozialem Kapital haftet also die Eigenschaft an, sein Einsatz nicht unabhängig von dem freien Willen der Community-Mitglieder zu planen[49]. Die Mitglieder einer Community können also nicht vertraglich gebunden werden. Ihnen kann also keine formale Rolle zugeordnet werden, wie dies bei Organisationen mit entsprechender Verfassung möglich ist[50]. Es

[48] vgl. Osterloh/Rota/Kuster, S. 6f.
[49] vgl. Osterloh/Rota/Kuster, S. 8
[50] vgl. Morner 2001, S. 5

existieren weder Sanktionsmöglichkeiten, noch Autorisierungsrechte im klassischen Sinne.

Daraus folgt, dass das Management dieses erfolgsrelevanten sozialen Kapitals einer besonderen Aufmerksamkeit bedarf. Um nun konkrete Handlungsempfehlungen für das Management des sozialen Kapitals aussprechen zu können, ist es notwendig, die Prinzipien und das Wesen von Communities zu beleuchten.

Auf den ersten Blick scheinen Open Source Communities dezentral und recht unkoordiniert organisiert zu sein[51]. So kann im Prinzip jedes Community-Mitglied das entsprechende OS-Programm nutzen, Änderungen am Quellcode vornehmen und diese Änderungen weitergeben. Die Praxis zeigt jedoch, dass eine gewisse Zentralität und Koordination dennoch vorhanden zu sein scheint. So ist beispielsweise beim OS-Projekt Debian[52] eine Zentralität in der Art und Weise feststellbar, dass tiefgreifende Aspekte der Weiterentwicklung in eigens eingerichteten Foren diskutiert werden (vgl. Mirau 2002, S. 26). Moderiert werden diese Foren von zwei übergeordneten Instanzen. Beim OS-Projekt Debian sind diese Prinzipien der Verhaltensregeln neben anderen Aspekten sogar in Statuten verankert[53]. Auch beim Open-Source Projekt Perl gibt es derartige Zentralitäten. So werden hier alle wichtigen Entscheidungen und dazugehörige Änderungen in diesem Projekt zunächst in einem festgelegten Phasenschema der Gruppendiskussion besprochen und anschließend darüber abgestimmt[54]. Ähnlich gehaltene zentrale Organisationsmuster sind bei fast allen OS-Projekten beobachtbar.

Auch das Merkmal der Koordination ist in OS-Projekten häufig zu beobachten. So sind die Projekte meist in Einzelbereiche gegliedert[55]. Das OS-Projekt Debian gliedert sich beispielsweise in 8000 Einzelbereiche, die von so genannten Maintainern betreut und geleitet werden[56]. Auch andere OS-Projekte lassen sich derart untergliedern. Es kann also durchaus von einer gewissen Organisationsstruktur in OS-Projekten gesprochen werden.

[51] vgl. Morner 2001, S. 5
[52] Erklären: siehe Motivation und Organisation von Open Source Projekten
[53] vgl. Debian Constitution 2003
[54] vgl. Morner 2001, S. 5; Markus/Manville/Agres 2000, S. 22
[55] vgl. Metiu/Kogut 2001; Mirau 2002, S. 25; Morner 2001, S. 5
[56] vgl. Mirau 2002, S. 25

Ein weiteres Merkmal, das auf Communities zutrifft, sind gemeinsam geteilte Werte und Einstellungen[57]. Neben gemeinsamen projektspezifischen Zielen verfolgen die Mitglieder einer Community auch soziale Werte und Einstellungen[58]. Als zentraler Aspekt dieser geteilten Werte und Einstellungen lässt sich die ablehnende Haltung gegen die wirtschaftliche Vermarktung von Software identifizieren. Mit dem Engagement in einer Community sprechen sich die Mitglieder gegen den Schutz geistigen Eigentums aus. So steht beim OS-Projekt GNU die Freiheit von Software als prinzipielles Ziel im Vordergrund[59].

Auch kann die Solidarität unter den Community-Mitgliedern als ein wesentliches Merkmal herausgestellt werden[60]. Die Solidarität beruht auf den gemeinsamen Werten und Einstellungen, aber auch auf dem gemeinsamen Projektziel, eine Software zu erstellen bzw. weiter zu entwickeln.

Das Phänomen der natürlichen Autorität ist ebenfalls ein Merkmal von OS-Communities. Eine natürliche Autorität wird in der Organisationstheorie demjenigen zugesprochen, der ein besonderes Wissen oder eine besondere Leistung vorweisen kann[61]. Natürliche Autorität erwächst also nicht aus einem Autorisierungsrecht, wie dies bei der verfassungsbedingten Autorität der Fall ist, sondern aus einem Expertentum[62]. Diese natürliche Autorität wird nicht selten bestimmten Schlüsselpersonen innerhalb von OS-Projekten zugesprochen, denen auf diese Weise ein nicht unerhebliches Beeinflussungspotenzial nachgesagt werden kann. Linus Torvalds, der Initiator des Linux-Projektes, stellt ein Beispiel dafür dar.

Wie bereits deutlich wurde, spielen für Community-Mitglieder geldliche Entlohnungsformen keine Rolle bei der Motivation, in einer Community mitzuwirken. Die Motive sind vielmehr auf ideeller Ebene zu finden. In der Literatur werden zahlreiche Motive identifiziert, die ebenso unterschiedlich wie zahlreich sind. Hier sollen die wichtigsten und für den vorliegenden Untersuchungsgegenstand benötigten

[57] vgl. Morner 2001, S. 6; Markus/Manville/Agres 2000, S. 17
[58] vgl. Morner 2001, S. 6
[59] vgl. Stallmann 1999
[60] vgl. Morner 2001, S. 6f.
[61] vgl. Ringlstetter 1997
[62] vgl. Morner 2001 S. 7

kurz vorgestellt werden. So stellt die Identifikation der Mitglieder mit der Community eine bedeutende Motivationskomponente dar[63]. Unter Community-Mitgliedern werden schnell soziale Bindungen eingegangen, die als starker Antrieb zur Mitwirkung in OS-Projekten gelten. Als Beispiel für eine starke Verbindung kann die Linux–Community genannt werden. Eine weitere Motivation stellt auch die Reputation der Mitglieder dar[64]. Durch hochqualifizierte Beiträge und Engagement steigt die Reputation einzelner Community-Mitglieder. Eine hohe Reputation wiederum stellt nicht nur an sich einen Wert dar, sondern kann auch positive Nebeneffekte mit sich bringen (zum Beispiel das Angebot für einen lukrativen Job). Auch die einfache Freude an der Programmierarbeit innerhalb von OS-Projekten kann eine Motivation darstellen[65]. Vergleichbar ist diese Freude mit der eines Menschen, der seinem Hobby nachgeht.

Die bis hier beschriebenen Eigenschaften von OS-Communities stellen die Grundlage für die Ausarbeitung von Handlungsempfehlungen für den Aufbau solcher Communities (soziales Kapital), auf die im Folgenden eingegangen werden soll.

Gestaltung einer Organisationsstruktur

Wie dargelegt, sind in Communities Ansätze einer strukturierten Organisation beobachtbar. Dieser Sachverhalt stellt für OS-Stellende einen Ansatzpunkt dar, die Geschicke der OS-Community zumindest grundlegend gemäß der eigenen Ziele zu beeinflussen, ohne dabei die Regeln und Grundsätze der OS-Idee zu verletzen.

Ausgangspunkt einer Gestaltung der OS-Community ist der Aspekt der natürlichen Autorität. Für das Gelingen dieses Vorhabens ist es nun von Vorteil, dass die Rolle der natürlichen Autorität dem OS-Stellenden zufällt. Als Initiator des OS-Projektes mit einem bis zur Initiierung des Projektes verfügten alleinigen Expertenwissens kommt dem OS-Stellenden diese Rolle quasi automatisch zu. Auf Grund der Glaubwürdigkeit und Kompetenz einer natürlichen Autorität steht dem OS-Stellenden der Weg offen, die Funktion eines Community-Gründers bzw. eine Moderators einer

[63] vgl. Lakhani 2002, S. 3; Mierau 2002; S. 11
[64] vgl. Lakhani/Hippel 2000, S. 3; Markus/Manville/Agres 2000, S. 15; Morner 2001, S. 7; Raymond 1999b
[65] vgl. Lakhani/Hippel 2000, S. 3; Mierau 2002, S. 17; Morner 2001, S. 7; Raymond 1999b

bestehenden Community einzunehmen und somit über ein Beeinflussungspotenzial zu verfügen.

Ferner sollte bei der Gründung oder Moderationsübernahme einer bestehenden Community dahingehend Einfluss genommen werden, dass das Projekt eine weitgehende Aufgabenteilung erfährt. Dieser Aspekt wird in der Literatur als wesentliche Voraussetzung für das erfolgreiche Gelingen des Projektes angesehen[66]. Auf diese Weise kann mit der meist vorhandenen Komplexität von OS-Projekten gerecht werden. Idealerweise sollte das Projekt also in kleinere Einheiten oder Module zerlegt werden, die von entsprechenden Teams bearbeitet werden.

In Anlehnung an die Arbeitsteilung sollten entsprechende Team-Hierarchien eingerichtet werden[67]. Wie oben beschrieben, sind derartige Hierarchien in OS-Projekten durchaus üblich. An der Spitze des Projektes sollte sich ein Kernteam etablieren, welchem der OS-Stellende als Initiator angehören sollte. Dieses Kernteam bestimmt wiederum Projektleiter für die oben angesprochenen Module, die für diese verantwortlich sind. Diese Projektleiter sollten eine eher moderierende Funktion als eine von oben herab bestimmende Funktion einnehmen (also keine Autorität), um den Konventionen von OS-Communities weitgehend zu entsprechen.

Um die Größe und die Qualität des Mitgliederstamms der OS-Community zu kontrollieren, hat sich das so genannte „Managed Membership“ bewährt[68]. Hierunter ist das bewusst gesteuerte Selektieren der Mitglieder zu verstehen. So müssen beispielsweise die Mitglieder des Apache-Projekts, bevor sie formal als solche anerkannt werden, eine Profilierungszeit absolvieren, in der sie ihre Qualifikationen einer Kerngruppe von Gründungsmitgliedern unter Beweis stellen[69]. Im Debian-Projekt wird die Mitgliederaufnahme alleine von entsprechenden Projektleitern bestimmt.[70]. Auf diese Weise lässt sich eine überschaubare Zahl an Mitgliedern hervor bringen, die zudem die richtige Qualifikation mitbringen. Ein Managed

[66] vgl. z.B. Lerner/Tirole 2000, S. 21; Markus/Manville/Agres 2000, S. 21; Morner 2001, S. 5
[67] vgl. Markus/Manville/Agres 2000, S. 21; Morner 2001, S. 5
[68] vgl. Markus/Manville/Agres 2000, S. 21; Morner 2001, S. 5
[69] vgl. Baylaws of The Apache Software Foundation 1999
[70] vgl. Debian Constitution 2003

Membership in ähnlicher Form sollte also auch in diesem Kontext Berücksichtigung finden.

Auch der Aspekt der Zentralität sollte bei der erfolgreichen Gründung einer OS-Community eine Rolle spielen. So wurde bereits auf bestimmte Abstimmungsverfahren in OS-Communities eingegangen. Um die Geschicke der OS-Projekte gemäß den eigenen Erwartungen und Wünschen lenken zu können, scheint die Implementierung derartiger Prozedere ein geeignetes Instrument zu sein.

Auch die Einführung bestimmter Verhaltensregeln für die Community-Mitglieder beeinflusst den Erfolg solcher Vorhaben. Solche Regeln können durch die Wahl eines geeigneten Lizenzmodells zum Tragen kommen. Dementsprechend sollte bei der Verfassung der Lizenz darauf entsprechend Rücksicht genommen werden. Allerdings sollte ein Regelwerk nicht zu komplex und unübersichtlich werden, damit ein zu ausgeprägter Formalismus sich nicht innovationshemmend auf das Projekt auswirkt.
Ein derartiges Regelwerk bedarf allerdings eines wiederum nicht zu formalistisch ausgestalteten Kontroll- und Sanktionsmechanismus. Diese Aufgabe sollte den Teamleitern zukommen

Makrokultur

In Gemeinschaften ist häufig eine Solidarität unter den Mitgliedern zu beobachten. Diese Solidarität beruht nicht zuletzt auf einer gemeinsamen Makrokultur. Eine Makrokultur ist ein System von fundamentalen Werten und Einstellungen der Mitglieder darüber, wie die Beziehungen der Mitarbeiter untereinander sowie zu Externen, Meinungen, Ideen, Ideale, die Rollen der Gemeinschaft und deren Mitglieder etc. spezifiziert sind. Die Makrokultur wird von allen Mitgliedern geteilt und akzeptiert. Auf diese Weise können Makrokulturen interdependente und sogar sehr komplexe Aktivitäten solcher Gemeinschaften koordinieren und fördern[71]. Das trifft auch und im besonderen Maße auf OS-Communities zu. Das kulturell geteilte Umfeld der Community macht eine erfolgreiche Zusammenarbeit möglich, auch wenn

[71] vgl. Jones/Hesterly/Borgatti 1997, S. 929 ff.

die Mitglieder nur virtuell miteinander verbunden sind[72]. Eine Makrokultur wirkt auch zusätzlich zu den oben erwähnten Mitteln als Kontroll- und Sanktionsinstrument. In einer gelebten Kultur herrscht ein sozialer Druck zur Einhaltung der eigenen Normen und Regeln[73]. So müssen Quertreiber den Unmut der anderen Mitglieder oder schlimmstenfalls mit den Ausschluss aus der Community fürchten.

Aus diesem Grund ist die Etablierung einer Kultur für das eigene OS-Projekt ratsam. Dabei muss aber berücksichtigt werden, dass dieses Vorhaben sehr schwierig in der Umsetzung sein kann. Da die entsprechenden Inhalte an sehr viele Mitglieder herangetragen und von diesen angenommen werden müssen, kann es unter Umständen Jahre dauern. Auch muss man sich über die speziellen Inhalte einer gemeinsamen Kultur im Klaren sein.

Motivation

Um die Community-Mitglieder für das Projekt zu begeistern bzw. zu gewinnen, müssen Motivationen geschaffen werden. Die oben beschriebenen Motivationen von Mitgliedern zur Beteiligung an OS-Projekten lassen sich allerdings nur bedingt von Seiten der Projektführung beeinflussen. So kann die Identifikation mit der Community sowie die Freude am Programmieren nur von den Projektmitgliedern selbst als Motivation aufgebracht werden. Einen Ansatzpunkt stellt aber die Reputation als Motivator dar. Zwar muss die Motivation von den Betroffenen immer noch selbst aufgebracht werden und hängt die Reputation von der Beurteilung Dritter ab, doch kann diese Art der Motivation indirekt beeinflusst werden. So könnten beispielsweise Auszeichnungen, Berichte in Zeitungen o.ä. die Entwicklung einer Motivation fördern. Dementsprechend sollte also interveniert werden.

12.4.2 Personelle Aspekte

Damit die zu gründende Community nicht zum unkontrollierten Eigenläufer wird und gänzlich gegen die Zielen der OSS veröffentlichenden Institution läuft, werden besondere Schnittstellen benötigt. Diese von der OSS veröffentlichenden Institution

[72] vgl. Markus/Manville/Agres 2000, S. 24
[73] vgl. Markus/Manville/Agres 2000, S. 23

bereitgestellten Schnittstellen stellen sicher, dass immer eine enge Verbindung mit der Community gewährleistet ist und eine Zielkonformität besteht. Allerdings soll hier nicht der Eindruck geweckt werden, eine Community mit hierarchischen Organisationsmustern leiten und führen zu müssen. Dies würde aus oben beschriebenen Gründen auch nicht funktionieren. Vielmehr sind hier Schnittstellen gemeint, die im oben gemeinten Sinne nutzbringende Beziehung zur Community pflegen. Dafür wurden in der OS-Literatur für derartige Verhältnisse von einer OSS veröffentlichenden Institution zu einer Community verschiedene Beziehungsmuster diskutiert. Behlendorf (1999) schlägt dazu fünf Rollen vor, die eine derartige Beziehung aufrechterhalten und pflegen. Da dieses Modell für den vorliegenden Untersuchungsgegenstand am Besten geeignet ist, soll es im Folgenden vorgestellt und empfohlen werden.

Rolle 1 – die Infrastrukturbetreuer

Es wird jemand benötigt, der die Adressen in den Mailinglisten, den Web-Server, den Quellcode-Server (CVS), die Datenbank mit den gefundenen Bugs, usw., pflegt.

Rolle 2 – der Code-„Käpt'n"

Diese Rolle hat einen Überblick über den Quellcode und ist für seine Qualität verantwortlich. Dazu muss sie neue Beiträge der Community implementieren, diese auf Bugs durchsuchen und auch selbst an Neuentwicklungen mitwirken.

Rolle 3 – der Bug-Verwalter

Die Entwickler-Gemeinschaft benötigt eine Plattform, auf der sie auf Fehler im Code aufmerksam machen können. Die Aufgabe des Bug-Verwalters ist die Bearbeitung der eingehenden Bug-Reports. Als „First-Level-Supporter" filtert er die einfachen Fragestellungen heraus, beseitigt die überflüssigen und leitet die anspruchsvollen an die Entwickler weiter.

Rolle 4 – der „Redakteur" (Dokumentation / Inhalt der Website)

Hierbei handelt es sich um eine wichtige, aber leider in vielen OS-Projekten vernachlässigte Aufgabe. Oft wird die Dokumentation direkt von den Programmierern

übernommen und dann nur halbherzig durchgeführt. Dabei ist dies gerade eine gute Möglichkeit das Projekt auch für nicht-technische Interessenten attraktiver zu gestalten und somit die Nutzung der neuen Entwicklungen auszuweiten. Ein positiver Nebeneffekt ist die Minderung der Bug-Reports, die teilweise nur Missverständnisse beinhalten, aber keine wirklichen Fehler. Zudem werden potenzielle Mitentwickler durch eine gute und verständliche Beschreibung des Programms eher dazu ermutigt sich mit der Entwicklungsumgebung auseinander zu setzen und später eventuell selber Beiträge zu leisten. Daher ist eine hochwertige Beschreibung der internen Architektur der Software unerlässlich und eine Erklärung der Hauptprozeduren mit Sicherheit förderlich. Außerdem obliegt dieser Rolle die Pflege der Projektwebsite.

Rolle 5 – der Organisator, Antreiber und Stratege

Diese Person schafft Impulse für das Projekt, indem sie neue Entwickler findet und an das Projekt heranführt, gezielt auf die Suche nach potenziellen Testanwendern geht oder Unternehmen findet, die die Entwicklung später einsetzen. Dieser Rolleninhaber sollte zwar mehr Verbundenheit zum technischen Bereich haben als der typische Vertriebler, aber er darf nie den Zweck des Projektes aus den Augen verlieren. Deswegen ist die Fähigkeit, Zusammenhänge zwischen einzelnen Bereichen sehen zu können ebenso von besonderer Bedeutung wie die Notwendigkeit, ein solches Projekt als Einheit zu betrachten. Zu Beginn muss sich der Stratege mit sämtlichen Teilbereichen des Projektes bekannt machen und die Gründe für einzelne Vorgehensschritte sowie deren Zusammenhänge verstehen.

Die hier vorgestellten Rollen sollten im Rahmen dieses Prozessschrittes implementiert werden, so dass sie entscheidend zum Erfolg einer Community-Gründung im Sinne der eigenen Ziele beitragen können. Wenn man die beanspruchte Zeit für die Wahrnehmung dieser Rollen im Rahmen eines kleineren Projektes schätzt, kommt man schnell auf drei Vollzeit-Arbeitsplätze. In der Realität werden häufig einige der beschriebenen Rollen von mehreren Personen ausgefüllt, die sich die Verantwortung in den Bereichen teilen. Bei einigen Projekten ist es auch ausreichend, wenn das Kern-Team durchschnittlich fünf Stunden pro Woche daran arbeitet, nachdem die größten Hindernisse beim Start des Projekts beseitigt worden sind. Zu Beginn eines solchen Vorhabens sollten sich die Initiatoren aber so intensiv darum kümmern, als würden sie ganz an einem Projekt ihres Arbeitgebers mitwirken.

13 Kontinuierliche Projektarbeit

Die in diesem Kapitel beschriebenen Aspekte der OS-Veröffentlichung berühren den in diesem Dokument beschriebenen Prozess der OS-Veröffentlichung nachhaltig. Die hier behandelten Aspekte zeigen auf, was nach dem Durchlauf des eigentlichen OS-Veröffentlichungssprozesses fortlaufend betrieben werden muss, um dem OS-Projekt ein langfristiges Überleben zu sichern. Während der eigentliche Prozess der OS-Veröffentlichung also einen einmaligen Initiierungscharakter aufweist, sind die im Folgenden beschriebenen Aspekte als kontinuierliche Herausforderungen zu verstehen.

13.1 Kontinuierliche Community-Betreuung

Diese Aufgabe kommt der OSS veröffentlichenden Institution zu, wenn in Schritt 7 der Weg der Eigendistribution gewählt worden ist (vgl. Schritt 7). Hierbei geht es darum, die gegründete Community langfristig am Leben zu erhalten. Dabei sollte die OSS veröffentlichende Institution eher eine zurückhaltend moderierende als eine autoritär führende Rolle einnehmen (vgl. Schritt 11). Wichtig ist, die Spielregeln einer OS-Community nicht zu verletzen. Schließlich soll das soziale Kapital, das von den mitwirkenden Community-Mitgliedern ausgeht, für die eigenen Interessen und Zielsetzungen genutzt werden.

Damit die initiierende Organisation die Moderatorenrolle einnehmen und den Kontakt zur Community wahren kann, sollten die in Schritt 11 beschriebenen Rollen besetzt werden. Auf diese Weise werden die nötigen Schnittstellen gebildet. Den Rolleninhabern obliegen dabei vor allem folgende Aufgabenschwerpunkte:

- Beiträge der Community implementieren
- Entwicklung des Programms dokumentieren und den Beteiligten zur Verfügung stellen
- Schnittstelle zwischen dem eigenen Team und den freiwilligen Entwicklern bilden
- Mitglieder akquirieren und motivieren
- neue Versionen veröffentlichen

Diese Aufgaben sind durch die in Schritt 11 vorgestellten Rolleninhaber wahrzunehmen, wobei je nach Umfang und Größe des Projektes mehrere Rollen von einer Person übernommen werden können.

Wie man erkennen kann, beinhalten diese fünf Aufgabenschwerpunkte keine Neuentwicklungen und Innovationen an der Software selbst. Wie beschrieben, ist dies jedoch auch nicht Sinn der Sache, da diese Aufgabe den Community-Mitgliedern obliegt. Die Rolleninhaber schaffen durch ihre „Instandhaltungen“, Verwaltungs- und Motivationsarbeiten vielmehr die Grundlage dafür, dass die Community den „schöpfenden“ Part in ihrem Projekt übernimmt.

13.2 Vermarktung von Zusatz- und Komplementärleistungen

Parallel zur Betreuung der Community sollte spätestens zu diesem Zeitpunkt auch der Start des Zusatzgeschäftes initiiert werden. Die Konkretisierungen und Planungen aus Schritt 4 sind also zu diesem Zeitpunkt umzusetzen. Wie im Schritt 4 erläutert, trifft dies auf Institutionen zu, die ihre OS-Software mit dem Ziel veröffentlichen, ein Zusatzgeschäft zu etablieren.

Da die eigentliche Vermarktung von Zusatz- und Komplementärleistungen das Vorgehensmodell selbst nicht weiter betrifft und dieses im weiteren Verlauf auch nicht mehr beeinflussen wird, ist dieser Schritt im Prozess einer OS-Veröffentlichung nur als Milestone zu verstehen, der den spätesten Zeitpunkt für den Start der Vermarktung empfiehlt. Die Vermarktung des Zusatzgeschäftes durchläuft einen eigenen Prozess, der nach betriebswirtschaftlichen Maßstäben zu beurteilen und bewerten sein wird.

13.3 Laufende Kontrolle des OS-Projektes

Ebenfalls zur laufenden Projektarbeit gehört die Projektkontrolle. Einer laufenden Projektkontrolle sollten sich diejenigen OS-Steller unterziehen, die sich für die Eigendistribution und damit der Projektbetreuung entschieden haben (vgl. Schritt 7). Diese Entscheidung wurde ja getroffen, weil entsprechende Zielsetzungen (vgl. Schritt 3) die Beibehaltung der Einflussnahmemöglichkeit bezüglich der Fortentwicklung der Software für angeraten erschienen ließen. Um rechtzeitig mögliche Interventionsmaßnahmen auf Grund der Strategie- und Zielfestlegung entgegenwirkenden Entwicklungen einleiten zu können bzw. den

Fortentwicklungsprozess generell überwachen zu können, bietet sich die laufende Projektkontrolle an.

Die hier vorgeschlagene Projektkontrolle sollte anhand der spezifischen Leistungen des OS-Projektes erfolgen. Bei dieser so genannten Leistungskontrolle soll anhand der Parameter Markt, Community und Softwarequalität beurteilt werden, ob das OS-Projekt gemäß den gesteckten Strategie- und Zielsetzungen verläuft.

Nachfolgend wird der Kontrollprozess der Parameter näher beschrieben:

Markt

Kontrollmaßstab des Parameters Markt sollte der Grad der Verbreitung der Software sein. Beispielsweise lässt sich anhand von Downloadstatistiken sehr schnell feststellen, ob die Anwenderzahl zugenommen hat. Da die Erhöhung der Anwenderzahl in der Zielhierarchie an oberster Stelle platziert ist (vgl. Schritt 3), kann diese Größe als Erfolgskriterium für die generelle Befolgung der OS-Strategie und der daraus resultierenden und erhofften Folgeeffekte angesehen werden.

Community

Bezüglich des Parameters Community sollte kontrolliert werden, ob ein Interesse von möglichen Mitgliedern besteht, sich in irgendeinem Ausmaß am Projekt zu beteiligen. Ein Beurteilungskriterium könnte hier die Zahl der mitwirkenden Mitglieder sein. Auch das Engagement der bestehenden Mitglieder könnte als Kriterium herangezogen werden. Gemessen könnte speziell dieses Kriterium beispielsweise an der Anzahl der eingereichten Beiträge eines Mitglieds. Neben den neuen Beiträgen der Community sollte man auch die eingehenden Bug-Reports unter die Lupe nehmen. Anhand der Anzahl und der Vielfältigkeit der erhaltenen Fehlerberichte lässt sich ebenfalls das Engagement ableiten.

Auf diese Weise sollte das Interesse und das Engagement der Mitglieder erfasst werden. Je nach Zielsetzung der OSS veröffentlichenden Institution stellt nämlich das soziale Kapital ein wichtiges Erfolgskriterium für das Gelingen der entsprechenden OS-Stratgie dar.

Qualität der Software

Ein weiterer Gegenstand der Kontrolle sollte das Software-Programm selbst sein. Ist es nach den eigenen Vorstellungen weiterentwickelt worden oder bleibt es hinter den Erwartungen zurück? Die Qualität der Weiterentwicklungen beeinflusst wesentlich den Erfolg der OS-Strategie. So könnte beispielsweise die Reputation Einbußen erleiden, falls die Softwarequalität im Rahmen des OS-Projektes abnimmt. Auch die Anwenderzahlen könnten in diesem Fall abnehmen.

Haben nun die Kontrollen innerhalb dieser drei Parameter Schwachstellen identifiziert, so sind in einem nächsten Schritt zunächst die Gründe hierfür zu ermitteln. So ist beispielsweise ein mangelndes Interesse von möglichen Community-Mitgliedern darauf zurückzuführen, dass die Spielregeln der OS-Bewegung nicht eingehalten worden sind. Beispielsweise hat Netscape bei der OS-Veröffentlichung seiner Browsersoftware Mozilla seinerzeit Lizenzbestimmungen festgelegt, die es Netscape ermöglichten, die Software wieder unter den Status einer proprietären Software zu setzen. Das hat die Community Mitglieder verärgert und von einem Engagement abgehalten (vgl. oben).

Ist die Ursache für die identifizierten Schwachstellen schließlich analysiert, so sind umgehend entsprechende Maßnahmen zur Behebung einzuleiten. Selbstverständlich gilt dies nur für die von der OSS veröffentlichenden Institution beeinflussbaren Schwachstellen.

Anhang

Anhang A: Fallbeispiel Zope[74], [75]

Die Firma Digital Creations brachte 1996 ein proprietäres Content-Management-System (CMS) unter dem Namen „Principia“ auf den Markt und versuchte, damit über Lizenzverkäufe Umsätze zu generieren. Allerdings ließ die Verbreitung dieser Software sehr zu wünschen übrig, was vor allem auf mangelnde Marketing-Budgets und dem geringen Bekanntheitsgrad der Firma zurück zu führen war. Dementsprechend blieben die Einnahmen aus den Lizenzgeschäften verschwindend gering, und so empfahl ein Investor der Firma schließlich, das Produkt unter einer Open-Source-Lizenz und unter der Bezeichnung "Zope" (Z-Object Publishing Environment) freizugeben, was 1998 auch geschah.

Der Erfolg war durchschlagend: Innerhalb kurzer Zeit bildeten sich Newsgroups, eine große Fangemeinde entstand und Gruppen von Entwicklern formierten sich, deren Mitglieder in alle Welt verstreut sind und sich untereinander über das Internet absprechen. Die Mutterfirma Digital Creations (die sich neuerdings nur noch Zope Corporation nennt) hat hierbei u.a. die Aufgabe der Moderation des Entwicklungsprozesses, d.h., sie ist für das "Einsammeln" von Veränderungen und Verbesserungen am Source Code zuständig und gibt offizielle Versionen heraus.

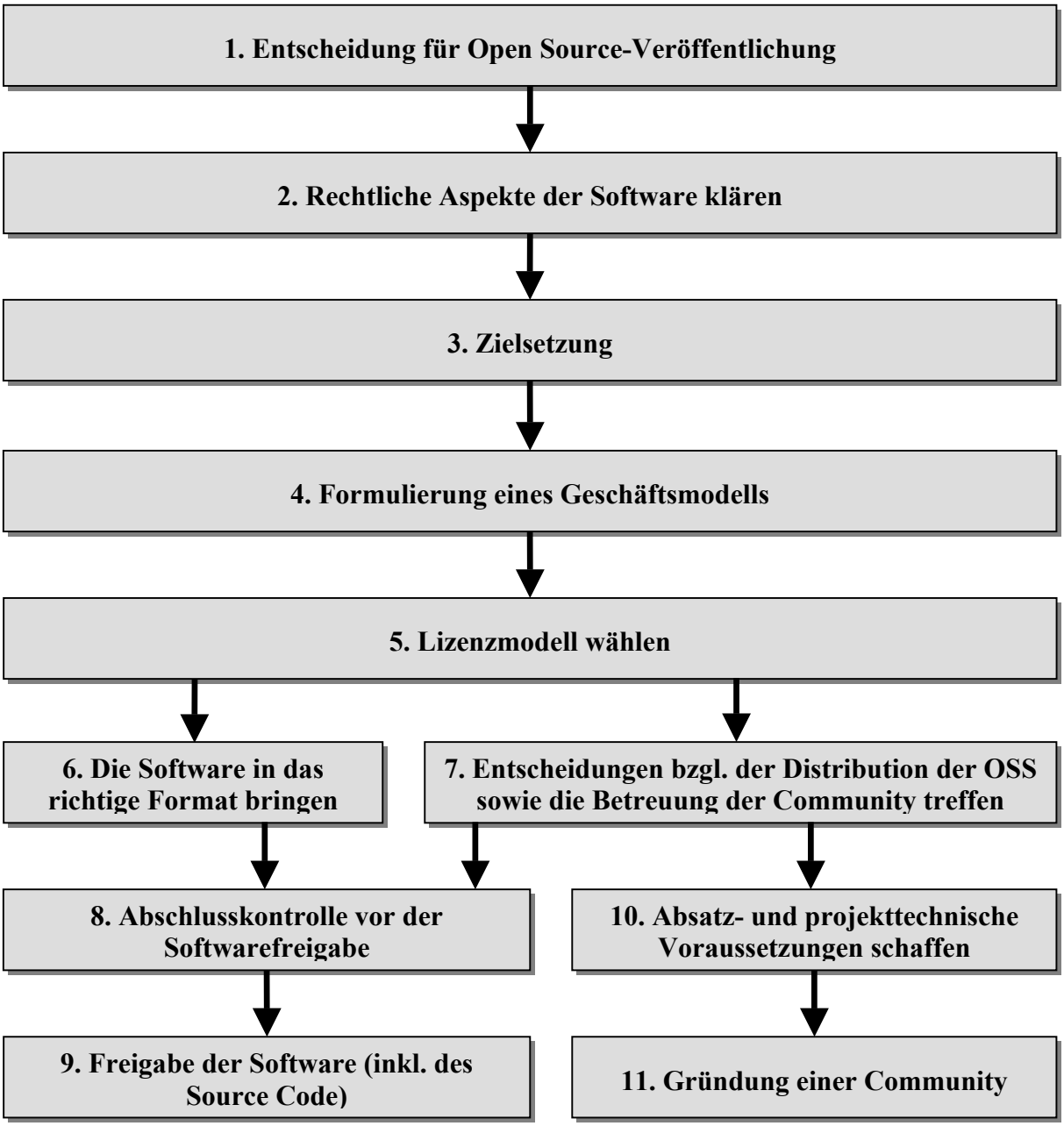
Das Unternehmen arbeitet allerdings nicht als non-profit-Organisation. Im Gegenteil, der Hauptgrund für die Freigabe des Programms war durchaus ein im kommerziellen Sinn pragmatischer: "... wenn die Anzahl der [...] Nutzer [von Zope] durch die Umwandlung in Open Source steigen würde, würde dies auch positive Auswirkungen auf die Haupteinnahmequellen von Digital Creations, nämlich Beratungs- und Entwicklungsdienstleistungen haben"[76]. Und so wird das Programm an sich zwar kostenlos zur Verfügung gestellt, im Bereich Installation und Service erwirtschaftet die Firma jedoch durchaus Gewinne durch Schulungen, Dokumentationen und die Entwicklung von Maßlösungen für Firmenkunden.

[74] vgl. Tippmann 2001 sowie Tegtmeyer 1999
[75] Beehive 2001
[76] Beehive 2001, S. 5

Ein wichtiger Hintergrund der großen Verbreitung von Zope liegt darin, dass die "Hemmschwelle" zur Benutzung sehr niedrig ist: Barrieren zur Installation und Verwendung von Zope gibt es praktisch nicht. Das ganze Paket hat einen Umfang von unter 5 MB, ist kostenlos zu downloaden und für alle gängigen Plattformen inklusive Windows 95/98 erhältlich (was recht selten ist; die meisten kommerziellen CMS benötigen mindestens eine "Professional-Version" von Windows oder ein anderes Server-fähiges System). Zudem sind alle Ausgabeprodukte von Zope plattformunabhängig und können auf jedem Betriebssystem eingesetzt werden.

Das Programmpaket selbst wird unter der eigens verfassten "Zope Public Licence" (ZPL) vertrieben, die zwar eine Open-Source-Lizenz ist, aber trotzdem die ökonomische Verwertbarkeit des Programms oder Teile davon für die Zope Corporation und auch für andere Firmen gewährleisten soll. Darin wird z.B. festgelegt, dass alle kommerziellen Programme, die mit Zope zusammen vertrieben werden oder damit zum Einsatz kommen, auch kommerziell und unter Copyright lizenziert bleiben dürfen. Ansonsten ist die ZPL eine nicht allzu strenge Open-Source-Lizenz, die nicht auf ganzer Linie den Idealen freier Software, wenn es nach der Free Software Foundation geht, entspricht und daher von dieser Stelle nur bedingt empfohlen wird.

Anhang B: Ablaufplan des OS-Veröffentlichungssprozesses



Anhang C: Checkliste für die OS-Veröffentlichung

1. Entscheidung für OS-Veröffentlichung

Umfeldanalyse	☐
Selbstanalyse	☐
Gegenalternativen aufstellen	☐
Strategisch günstigste Alternative wählen	☐

2. Rechtliche Aspekte der SW klären

Urheberrechtliche Aspekte der Software klären	☐
Verschiedene Lizenzbestimmungen berücksichtigen	☐
Haftungs- und Gewährleistungsfunktionen beachten	☐

3. Zielsetzung

Handlungsziel aus Strategie ableiten	☐
Ober- und Unterziele definieren	☐
Zielkonformität sicherstellen	☐

4. Formulierung eines Geschäftsmodells

Markt

Zielgruppe und Marktsegmente für Zusatzleistungen bestimmen	☐
Preise für die Zusatzleistungen ermitteln	☐
Distributionspolitische Aspekte klären	☐
Kommunikationspolitische Aspekte klären	☐

Gewinnmuster

Verkaufsfähige Leistungen / Produkte ausarbeiten	☐
Weitere produktpolitische Aspekte klären	☐

Ressourcenfokus

Personelle Ressourcen klären	☐
Sachliche Ressourcen klären	☐
Finanzielle Ressourcen klären	☐
Eventuell fehlende Ressourcen beschaffen	☐

Geschäftsstrategie

Bedrohungspotenziale erkennen	☐
Strategische Absicherung aufbauen	☐

Unternehmenskultur / Organisation / Mitarbeiter

Organisationssystem anpassen bzw. entwickeln	☐
Unternehmenskultur anpassen	☐
Mitarbeiter motivieren	☐

5. Lizenzmodell wählen

Standardlizenz auswählen	☐
Standardlizenz an eigene Bedürfnisse anpassen	☐
oder	
Eigene Lizenz entwerfen	☐

6. Das richtige Format für die Software finden

Eigene Versionsnummer für das OS-Release festlegen	☐
Entsprechende Dokumentationen bereitstellen	☐
Source Code in einfache und klare Struktur bringen	☐
Distribution anwenderfreundlich machen	☐

7. Entscheidung bzgl. der Distribution der OSS sowie die Betreuung der Community treffen

Entscheidung über Eigen- oder Fremddistribution fällen	☐

8. Abschlusskontrolle vor der Softwarefreigabe

Rechtlichen Reviewprozess durchlaufen	☐
Technischen Reviewprozess durchlaufen	☐
Betriebswirtschaftlichen Reviewprozess durchlaufen	☐
Kompatibilität der Teilergebnisse prüfen	☐

9. Freigabe der Software

Softwarefreigabe	☐

11. Absatz- und projekttechnische Voraussetzungen schaffen

Website einrichten	☐
Downloadbereich auf der Website zur Verfügung stellen	☐
Informationen über das Projekt auf der Website bereitstellen	☐
Entsprechende Kommunikationskanäle auf der Website einrichten	☐
CVS installieren	☐

12. Gründung einer Community

Soziale Aspekte:

Organisationsstruktur gestalten	☐
Makrokultur etablieren	☐
Motivationen schaffen	☐

Personelle Aspekte:

Schnittstellen einrichten	☐

Anhang D: Kompetenzeinschätzung für eine Open-Source-Veröffentlichung

Schritt	Berührte Kompetenzen	
1. Entscheidung für eine OS - Veröffentlichung	Technisch	
	Rechtlich	
	Betriebswirtschaftl.	●●●●●
Der Prozessschritt der Entscheidungsfindung berührt alleine die betriebswirtschaftlichen Kompetenzen. Die hier zu fällenden strategischen Entscheidungen betreffen konkret das Management der höheren Ebenen einer Institution, weil eine derartige Strategieentscheidung von grundlegender Natur ist. Eine Aufwandseinschätzung gemäß der angestrebten Punkteverteilung ergibt sich deshalb wie oben.		
2. Rechtliche Aspekte der Software klären	Technisch	
	Rechtlich	●●●●●
	Betriebswirtschaftl.	
Diese Aufgabe bedarf vor allem rechtliches Know-how und Hintergrundwissen und sollte dementsprechend von Experten auf diesem Gebiet erfüllt werden. So sollte diesbezüglich unbedingt der juristische Rat der Rechtsabteilung oder von unabhängigen Rechtsanwälten eingeholt werden. Eine Punkteverteilung ergibt sich deshalb einheitlich zu Gunsten der rechtlichen Kompetenzen.		
3. Zielsetzung	Technisch	●
	Rechtlich	
	Betriebswirtschaftl.	●●●●
Die Zielformulierung für die OS-Veröffentlichung wird in erster Linie von betriebswirtschaftlichen Kompetenzen bestimmt. Im Detail wird dies sehr wahrscheinlich vom strategischen Management in einer konkreten Formulierung und Einordnung der Ziele in die Zielhierarchie bzw. des Zielsystems der Institution enden. Für die Formulierung der Ziele bedarf es allerdings eines technischen Hintergrundwissens. So muss von technischer Seite geklärt werden, ob potenzielle Ziele technisch umsetzbar sind. Eine Aufwandseinschätzung ergibt deshalb nach obigem Bild.		
4. Formulierung eines Geschäftsmodells	Technisch	
	Rechtlich	
	Betriebswirtschaftl.	●●●●●

In diesen Prozessschritt wird vor allem die strategische Planungseinheit der entsprechenden Institution involviert sein. Diese liegt meist beim Management bzw. der mittleren bis oberen Führungsebene. Wie die Ausführungen gezeigt haben, sind dafür alleine betriebswirtschaftliche Kompetenzen gefragt. Neben den organisatorischen Tätigkeiten, wie z.B. die Erarbeitung der Geschäftsstrategie, die Bestimmung der Zielgruppe und dem Erstellen von Zusatzangeboten, spielt hier aber auch der Finanzbereich eine Rolle. Seine Aufgabe ist es, einen Finanzierungsplan für das Vorhaben zu erarbeiten. Eine Punkteverteilung ergibt sich zu Gunsten der betriebswirtschaftlichen Kompetenzen.

5. Lizenzmodell wählen	Technisch	
	Rechtlich	●●●●
	Betriebswirtschaftl.	●

Die lizenzrechtlichen Fragen dieses Prozessschrittes dürften in erster Linie die rechtlichen Kompetenzen fordern. Wie die obigen Ausführungen gezeigt haben, ist bei der Formulierung der Lizenzen auf die Daten der Zielsetzung und Strategieformulierung zurückzugreifen. Deshalb sind für diesen Schritt auch die betriebswirtschaftlichen Kompetenzen gefragt. Eine Punkteverteilung sieht entsprechend wie oben aus.

6. Die Software in das richtige Format bringen	Technisch	●●●●●
	Rechtlich	
	Betriebswirtschaftl.	

In der Umsetzung dieses Prozessschrittes werden vor allem technische Kompetenzen berührt. Die beschriebenen Aufgaben sollten idealerweise von den ursprünglichen Programmierern der Software erfüllt werden, da bei ihnen das entsprechende Software-Know-how liegt. Das Bild der Punkteverteilung sieht daher wie oben aus.

7. Entscheidung über Distribution der OSS und Betreuung der Community treffen	Technisch	●●◖
	Rechtlich	
	Betriebswirtschaftl.	●●◖

Dieser Prozessschritt benötigt für seine Umsetzung sowohl technisches als auch betriebswirtschaftliches Know-how. Es werden auf technischer Seite konkret Kenntnisse im Umfeld des Internets und des CVS benötigt. Die betriebswirtschaftlichen Kompetenzen sollten konkret aus dem Marketingbereich sowie des strategischen Managements kommen. Eine Punkteverteilung ergibt sich deshalb zu Gunsten der technischen und betriebswirtschaftlichen Kompetenzen.

8. Abschlusskontrolle vor der Softwarefreigabe	Technisch	●◖
	Rechtlich	●◖
	Betriebswirtschaftl.	●●

Die Erfüllung der Aufgaben erfordert alle Kompetenzen. So obliegt es der rechtlichen Kompetenz alle rechtlichen Aspekte zu kontrollieren. Entsprechendes gilt für die technischen und betriebswirtschaftlichen Aufgabenträger. Bei der Punkteverteilung soll der betriebswirtschaftlichen Kompetenz allerdings ein größeres Gewicht zukommen, da der Kontrollaufwand für diesen Kompetenzbereich entsprechend höher ausfallen dürfte.		
9. Freigabe der Software (inkl. des Source Code)		
Milestone		
10. Absatzt- und projekttechnische Voraussetzungen schaffen	Technisch	●●●●●
	Rechtlich	
	Betriebswirtschaftl.	
Dieser Prozessschritt bedarf vor allem technische Kompetenzen. Entsprechend sollten hier IT-Fachkräfte mit Erfahrungen im Internetbereich eingesetzt werden. So ergibt sich eine Punkteverteilung gemäß der obigen Abbildung.		
11. Gründung einer Community	Technisch	
	Rechtlich	
	Betriebswirtschaftl.	●●●●●
Dieser Prozessschritt erfordert betriebswirtschaftliche Kompetenzen. Mögliche Kompetenzträger sollten dabei sowohl aus den Teildisziplinen Personalwirtschaft und Organisationstheorie stammen. Es ergibt sich dementsprechend obiges Bild.		

Referenzverzeichnis

AGB (1976): Gesetz zu Regelung des Rechts der Allgemeinen Geschäftsbedingungen (AGB-Gesetz), veröffentlicht im Internet: URL: http://www.nonprofit-management.de/gesetze/agbg/0f.htm [Stand: 17.10.2003].

Baylaws of The Apache Software Foundation (1999): Baylaws of The Apache Software Foundation, veröffentlicht im Internet: URL: http://www.apache.org/foundation/bylaws.html [Stand: 11.09.2003].

Beehive GmbH (Hrsg.) (2001): ZOPE: Content-Management- & Web-Application-Server, Heidelberg: dpunkt.verlag, 2001.

Behlendorf, B. (1999): Open Source as a Business Strategy, in: DiBona, Ch.; Ockman, S.; Stone, M. (Hrsg.): Open Sources: Voices from the Open Source Revolution, veröffentlicht im Internet: URL: http://www.oreilly.com/catalog/opensources/book/toc.html [Stand: 11.09.2003].

Bieger, T.; Rüegg-Stürm, J.; Rohr, T. von (2002): Strukturen und Ansätze einer Gestaltung von Beziehungskonfigurationen – Das Konzept Geschäftsmodell, in: Bieger, T. et al. (Hrsg.): Zukünftige Geschäftsmodelle: Konzepte und Anwendungen in der Netzökonomie, Berlin et al.: Springer, 2002, S. 35 – 61.

BGB (1998): Bürgerliches Gesetzbuch (BGB), veröffentlicht im Internet: URL: http://www.nonprofit-management.de/gesetze/bgb/Of.htm [Stand: 17.10.2003].

DBUS (o. J.): DBUS! Deutschland, veröffentlicht im Internet: URL: http://www.dbus.de/eip/inhalt.html [Stand: 11.09.2003].

Debian Constitution (2003): Debian Constitution: Constitution for the Debian Project (v1.1), veröffentlicht im Internet: URL: http://www.debian.org/devel/constitution [Stand: 11.09.2003].

GPL (1991): General Public License, veröffentlicht im Internet auf der Seite der Free Software Foundation: URL: http://www.fsf.org/copyleft/gpl.html [Stand: 17.10.2003].

Grassmuck, V. (2002): Freie Software zwischen Privat- und Gemeineigentum, Bonn: Bundeszentrale für politische Bildung (bpb), 2002.

Hang, J.; Hohensohn, H.(2003): Eine Einführung zum Open Source Konzept aus Sicht der wirtschaftlichen und rechtlichen Aspekte, veröffentlicht im Internet: URL: http://www.c-lab.de/home/de/reports/index.html [Stand: 21.07.2003]

Jaeger, T. (2000): GPL und Haftung: Ohne Verantwortung?, veröffentlicht im Internet: URL: http://www.ifross.de/ifross_html/art3.html [Stand: 16.09.2003]

Jones, C.; Hesterly, W. S.; Borgatti, S. P. (1997): A General Theory of Network Governance: Exchange Conditions and Social Machanism, in: Academy of Management Review, Oct. 1997, Vol. 22, Iss. 4, S. 911 – 945.

Lakhani, K. (2002): The Boston Consulting Group Hacker Survey, veröffentlicht im Internet: URL: http://www.osdn.com/bcg/BCGHACKERSURVEY-0.73.pdf [Stand: 20.09.2002].

Lakhani, K.; Hippel, E. v. (2000): How Open Source software works: “Free” user-to-user assistance, Working Paper No. 4117, MIT Sloan School of Management, May 2000.

Leiteritz, R. (2002): Der kommerzielle Einsatz von Open Source Software und kommerzielle Open Source-Geschäftsmodelle, veröffentlicht im Internet: URL: http://www.ig.cs.tu-berlin.de/ap/rl/2002-05/Leiteritz-DA-OSS-Geschaeftsmodelle-052002.pdf [Stand: 13.09.2003].

Lerner, J.; Tirole, J. (2000): The Simple Economics of Open Source, Working Paper No. 7600, National Bureau of Economic Research, Cambridge, December 2000.

Lizenz Center (o. J.): Auflistung der Lizenzen; veröffentlicht im Internet: URL: http://www.ifross.de/ifross_html/lizenzcenter.html [Stand: 12.09.2003].

Markus, M. L.; Manville, B.; Agres, C. E. (2000): What Makes a Virtual Organization Work?, in: Sloan Management Review, Fall 2000, Vol. 42, Iss. 1, S. 13 – 26.

Marx, B. (2001): Linux Manager Guide: Was Entscheider über das Betriebssystem und Open Source wissen müssen, Nürnberg: SuSE-Press, 2001.

Metiu, A; Kogut, B. (2001): Distributed Knowledge and the Global Organization of Software Development, veröffentlicht im Internet:
URL: http://opensource.mit.edu/papers/kogut1.pdf [Stand: 10.09.2003].

Mierau, C. C. (2002): Motivation und Organisation von Open Source Projekten, veröffentlicht im Internet:
URL: http://www.medienkultur.org/sm2/org/tragik/ha/projektarbeit_opensource.pdf [Stand: 10.09.2003].

Morner, M. (2001): Das Open-Source-Software-Phänomen: Betrachtung aus organisationstheoretischer Perspektive: Diskussionsbeitrag zum Forschungsseminar in Westendorf, veröffentlicht im Internet:
URL: http://www.ku-eichstaett.de/Fakultaeten/WWF/Lehrstuehle/OP/Downloads/papers/Sections/content/mim_opensource.pdf [Stand: 10.09.2003].

Nehls, R. G., Baumgartner, P. (2000): Value Growth: Neue Strategieregeln für wertorientiertes Wachstum – Ein Ansatz von Mercer Mangement Consulting, in: Management Consulting, München, S. 79 – 93.

OSD (2003): The Open Source Definition, Version 1.9, veröffentlicht im Internet:
URL: http://www.opensource.org/docs/definition.php [Stand: 16.10.2003].

o.V. (2003a): Angriff auf Windows, in: manager-magazin.de, März 2003, veröffentlicht im Internet: URL: www.manager-magazin.de/ebusiness/cebit/0,2828,240744,00.html, [Stand: 18.08.2003].

o.V. (2003b): RedHat meldet Gewinn, in: pro-linux, Juni 2003, veröffentlicht im Internet: URL: www.pro-linux.de/news/2003/5642.html, [Stand: 19.06.2003].

o.V. (2003c): Open-Source-Lizenzen: OpenFacts, die Open-Source-Wissensdatenbank, veröffentlicht im Internet: URL: http://openfacts.berlios.de/index.phtml?title=Open-Source-Lizenzen [Stand: 09.04.2003].

o.V. (1999): Open Source - kurz & gut, Sebastopol et al.: O'Reilly, 1999.

Osterloh, M.; Rota, S.; Kuster, B. (2002): Die kommerzielle Nutzung von Open Source Software: Der Einfluss von sozialem Kapital, in: Zeitschrift Führung und Organisation (ZFO), (2002), Nr. 4, S. 211 – 217.

Raymond, E. S. (1999a): Der verzauberte Kessel, veröffentlicht im Internet: URL: http://www.oreilly.de/opensource/magic-cauldron/cauldron.g.01.html [Stand: 10.09.2003].

Raymond, E. S. (1999b): Die Kathedrale und der Basar, veröffentlicht im Internet: URL: http://gnuwin.epfl.ch/articles/de/Kathedrale/catb_g.0.html#toc10 [Stand: 06.09.2003].

Raymond, E. S.; Lohmeier, M. (2001): Software Release Praxis HOWTO, veröffentlicht im Internet: URL: http://gd.tuwien.ac.at/opsys/linux/DE-HOWTO/HOWTO-test/DE-Software-Release-Praxis-HOWTO.html [Stand: 11.09.2003].

Ringlstetter, M. J. (1997): Organisation von Unternehmen und Unternehmensverbindungen, München et al.: Oldenbourg 1997.

Schaufler, E. (2000): Open Source Software Development: Kooperations- und Koordinationsmechanism, Seminararbeit an der Wirtschaftsuniversität Wien, veröffentlicht im Internet: URL: http://wwwai.wu-wien.ac.at/~koch/lehre/inf-sem-ss-00/schaufler/seminaross.html [Stand: 09.09.2003].

Schiffner, T. (2003): Open Source Software: Freie Software im deutschen Urheber- und Vertragsrecht, München: VVF Verlag, 2003

Schmitz, P.-E. (2001): Study into the use of Open Source Software in the Public Sector, veröffentlicht im Internet: URL: http://europa.eu.int/ISPO/ida/export/files/en/840.pdf [Stand: 12.09.2003].

Schönfeldt, R.; Raison, A. v. (2001): Linux im Unternehmen: 2. Business-Kongress auf dem LinuxTag, Heidelberg: dpunktverlag, 2001.

Schreyögg, G. (1984): Unternehmensstrategie: Grundfragen einer Theorie strategischer Unternehmensführung, Berlin und New York: Gryter, 1984.

Siepmann, J. (1999): Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software, veröffentlicht im Internet: URL: http://www.kanzlei-siepmann.de/linux-tag/vortrag.html [Stand: 10.09.2003].

Stallmann, R. (1999): The GNU Operating System and the Free Software Movement, in: DiBona, Ch.; Ockman, S.; Stone, M. (Hrsg.): Open Sources: Voices from the Open Source Revolution, veröffentlicht im Internet: URL: http://www.oreilly.com/catalog/opensources/book/toc.html [Stand: 11.09.2003].

Steinmann, H.; Schreyögg, G (2000): Management: Grundlagen der Unternehmensführung: Konzepte – Funktionen – Fallstudien, 5. Aufl., Wiesbaden: Gabler, 2000.

Tegtmeyer, F. (1999): Wie und warum Zope Open Source wurde, in: iX Magazin für professionelle Informationstechnik, veröffentlicht im Internet: URL: http://www.heise.de/ix/artikel/1999/08/064/01.shtml [Stand: 05.08.2003].

Tippmann, D. (2001): Open Source und Zope - Eine Einführung in Freie Software, veröffentlicht im Internet: URL: http://userpage.fu-berlin.de/~danitipp/daniel/opensource.html#Einfuhrung [Stand: 04.08.2003].

Weber, N.; Suhr, G. (2001): Ausarbeitung: Rechtliche Aspekte von OSS, veröffentlicht im Internet: URL: http://flp.cs.tu-berlin.de/lehre/wise00-01/oss/seminar/law_paper.html [Stand: 23.10.03].

Werth, S. (2003): Open Source Beziehungen – Urheberschaft / Schenkung oder gesellschaftlicher Ansatz, in: C-LAB Report, Vol. 2, Nr. 5, 2003, veröffentlicht im Internet: URL: http://www.c-lab.de/home/de/reports/index.html [Stand: 06.11.2003].

[77] vgl. Osterloh/Rota/Kuster,2002
[78] vgl. Grassmuck 2002, S. 307 ff.
[79] vgl. Raymond/Lohmeier 2001

Cooperative Computing &
Communication Laboratory

C-LAB
Marketing
Fürstenallee 11
D-33102 Paderborn

Telephone ++49-5251-60-6060
Telefax ++49-5251-60-6066
E-Mail marketing@c-lab.de
URL http://www.c-lab.de

Befragung über Ihre Zufriedenheit mit dem Report *„Open Source Software-Release“*

Wir bitten Sie, sich einen kurzen Moment Zeit zu nehmen, um uns ein paar Fragen über Ihre Einschätzung dieses Reports zu beantworten. Damit helfen Sie uns, Ihre Bedürfnisse besser zu verstehen. Wir möchten unsere Reports stärker nach Ihrem Interesse ausrichten, um so einen größeren Mehrwert bieten zu können. Vielen Dank für Ihre Mitarbeit.

Wie bewerten Sie das Thema dieses Reports?

	trifft voll zu			trifft überhaupt nicht zu
Aktuell	☐	☐	☐	☐
Interessant	☐	☐	☐	☐

Wie bewerten Sie den Inhalt dieses Reports?

	trifft voll zu			trifft überhaupt nicht zu
Aktuell	☐	☐	☐	☐
Interessant	☐	☐	☐	☐
Verständlich	☐	☐	☐	☐
Praxisrelevant	☐	☐	☐	☐
Informativ	☐	☐	☐	☐
Innovativ	☐	☐	☐	☐

Weitere Kommentare:

_____
_____
_____
_____

Freiwillige Angaben:
Name, Vorname: _____
Telefon: _____
E-Mail: _____

Bitte senden Sie das ausgefüllte Formular per Post, Fax oder E-Mail an die jeweilige Adresse (s. o.).

Kooperation
Siemens Business Services GmbH & Co. OHG
Universität Paderborn

Universität Paderborn
Prof. Dr. Franz-Josef Rammig
Telephone ++49-5251-606062

Siemens Business Services GmbH & Co. OHG
Dr. Wolfgang Kern
Telephone ++49-5251-606061