Open Source - Produktion, Organisation und Lizenzen

Markus Nüttgens, Enrico Tesei

Heft 157

Hrsg.: Prof. Dr. Dr. h.c. A.-W. Scheer
Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi),
Universität des Saarlandes, Im Stadtwald, Gebäude 14.1, D - 66123 Saarbrücken,
phone: (+49) 681-302-3106, fax: (+49) 681-302-3696,
email: iwi@iwi.uni-sb.de

Saarbrücken, Januar 2000

Inhaltsverzeichnis

Release early.

Release often.

And listen to your customers.

Eric Raymond

1 Open Source - Produktion

1.1 Entwicklungsmodell

Nach Eric Raymonds[1] Essay „The Cathedral and the Bazaar“[2] wird das Software-Entwicklungsmodell auch Basarmethode genannt. Eric Raymond hat bei seiner Analyse des Software-Entwicklungsmodells von Linux folgende Regel aufgestellt:

„Veröffentliche früh und häufig, delegiere alles was sich delegieren läßt und sei offen bis zum Punkt des heillosen Durcheinanders, genannt Chaos.“[3]

Weiterhin wurden folgende Faustregeln von ihm festgelegt:

- Jedes gute Programm hat seinen Ursprung in einer für den Entwickler störenden Unzulänglichkeit.
- Gute Programmierer wissen, was geschrieben werden muß. Große Programmierer wissen, was neu geschrieben werden muß und was wiederverwendet werden kann.
- Plane etwas zu verwerfen, denn du wirst es so oder so tun.
- Wenn man die richtige Einstellung hat, sind es die interessanten Probleme, die einen finden.
- Wenn man das Interesse an einem Programm verloren hat, dann besteht die letzte Pflicht in der Weitergabe an einen kompetenten und fachkundigen Nachfolger.
- Seine Benutzer als Mitentwickler einzubeziehen, ist die einfachste Art, den

[1] Vgl. Raymond, E. (Hrsg.): Eric Steven Raymond's Home Page: <URL: http://www.tuxedo.org/~esr/>, online: 06.10.99.

[2] Vgl. Raymond, E. (Hrsg.): The Cathedral and the Bazaar, <URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>, online: 23.09.99.

[3] Raymond, E. (Hrsg.): The Cathedral and the Bazaar, <URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>, online: 23.09.99.

Programmcode schnell zu verbessern und die Effizienz der Fehlersuche zu steigern.

- Veröffentliche früh, oft und höre auf die Kunden.
- Ist die Basis an Mitentwicklern und Beta-Testern ausreichend groß, dann wird nahezu jedes Problem schnell charakterisiert werden und die Lösung jemand offensichtlich sein.
- Debugging (Austesten und Fehlerbeseitigen) ist parallelisierbar.
- Wenn man seine Tester als die wertvollste Hilfsquelle behandelt, werden sie als eine solche reagieren.
- Nach guten eigenen Ideen ist das Erkennen guter Ideen anderer am besten. Manchmal ist letzteres aber noch besser.
- Oft kommen die besten und innovativsten Lösungen durch die Erkenntnis, daß der Lösungsansatz und das daraus resultierende Konzept falsch waren.
- Perfektion (im Design) ist nicht erreicht, wenn nichts mehr hinzuzufügen ist sondern wenn nichts mehr entfernt werden kann.
- Vorausgesetzt, der Projektleiter hat als Hilfsmittel ein Kommunikationsmedium mindestens von der Qualität des Internets und er versteht es ohne Zwang zu führen, sind viele Köpfe unvermeidlich besser als einer.

Die Vorgehensweise bei der Entwicklung freier Software war bis 1997 nicht dokumentiert und ausschließlich an bekannten freien Software-Projekten, wie z. B. Linux ersichtlich. Eric Raymond hat im Mai 1997 erstmalig versucht, den Erfolg einer Vorgehensweise anhand vieler erfolgreicher Projekte zu dokumentieren und zu veröffentlichen. Dies ist auch bis heute die einzige derartige Veröffentlichung.

1.2 Distributionsmodell

Entwickler stellen die Software im Internet für Benutzer zur Verfügung. Die Benutzer können selbst nach Software suchen, diese testen und einsetzen. Ein Benutzer kann per E-Mail in direkten Dialog mit Entwicklern treten und an der Entwicklung der Software teilnehmen, indem er die Software beurteilt und diese Beurteilung an die Entwickler sendet. Eine weitere Möglichkeit besteht darin, Bündel zusammengestellter Software bei einem Software-Händler zu kaufen. Distributoren, wie beispielsweise SuSE, Red Hat, Turbo Linux oder Debian bieten in der Praxis Software-Bündel an. Im Unterschied zu proprietärer Software bezahlt man für

freie Software ausschließlich für den Service der Bereitstellung, während bei proprietärer Software Lizenzgebühren zu entrichten sind. Konkret bedeutet das, daß eine einmal erworbene freie Software nach belieben verteilt werden kann. Die Distributoren stelle eigenständig Software-Pakete zusammen und verteilen diese gebündelt an Benutzer. Weiterhin hat der Benutzer die Möglichkeit, freie Software direkt vom Entwickler zu beziehen. Distributionsformen sind hauptsächlich Download-Möglichkeiten via Internet und der CD-Versand, der entweder direkt nach schriftlicher, elektronischer oder telefonischer Bestellung erfolgt oder indirekt indem CDs als Beilage in Fachzeitschriften und Sonderheften oder als Messe- und Werbegeschenke verteilt werden.

2 Open Source - Organisation

Unternehmen klagen einerseits über die Verschwendung von Ressourcen, beispielsweise durch unnötige Doppelarbeit, organisatorischen Leerlauf, zu hohe Gemeinkosten usw. und andererseits über mangelnde Flexibilität, zu langsame Anpassung an Umweltveränderungen und Fehleranfälligkeit verschiedener unternehmensinterner Systeme. Zur Problemlösung fallen in diesem Zusammenhang die Begriffe Organizational Slack, Redundanz und lose Kopplung.[4]

- *Redundanz* bedeutet Überfluß, Wiederholung oder Duplizierung von Informationen oder Aufgaben, um die Verständlichkeit, Vollständigkeit und Sicherheit einer Nachrichtenübermittlung oder Aufgabeerfüllung zu gewährleisten.
- *Organizational Slack*, im weiteren einfach *Slack* genannt, meint Ressourcenüberschuß in einer Organisation über das Notwendige zur Zielerreichung hinaus. In dieser Betrachtung bedeutet Slack, daß eine Unternehmung mehr Anreize bereitgestellt als zur Zielerreichung notwendig wäre. Slack wird, abgesehen von Ausnahmen wie der Bildung von Reserven, nur dann eingesetzt, wenn es um die Entwicklung von neuen Ideen und Innovationen geht oder um Lernen im weitesten Sinne. Dabei ist die Ressource Zeit, die in Unternehmen ohne Slack knapp ist, die entscheidende Erfolgsvariable und wird vor allem in Unternehmen der Software-Branche als einer der wichtigsten Gründe für

[4] Vgl. Staehle, W. H.: Redundanz, Slack und lose Kopplung, in: Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 313-345.

Wettbewerbsvorteile gegenüber Konkurrenten betrachtet.

- *Lose Kopplung* erlaubt im Gegensatz zur engen, starren Verkettung die lockere Verknüpfung teilautonomer Gruppen, Abteilungen oder Systemen. Fehler und Störungen eines Teilsystems sollen so nicht mehr auf das Gesamtsystem durchschlagen, sondern dezentral behoben werden können.

Die drei Konzepte Redundanz, Slack und lose Kopplung stammen zum Teil aus verschiedenen Disziplinen, beispielsweise aus der Biologie, den Ingenieurwissenschaften, der Kybernetik, der Systemtheorie und der Soziologie. Indem sie mit Verschwendung und unwirtschaftlichem Ressourcenumgang in Verbindung gebracht werden können sie negativ belegt sein. Es kommt jedoch auf die Nutzung der Überschußressourcen an. Bleiben sie ungenutzt, dann ist dieser Slack negativ zu bewerten. Werden sie als Potentiale für die Bearbeitung von Aufgaben genutzt, dann ist dies positiv zu bewerten. Was heute als Ressourcenverschwendung erscheint, kann in Zukunft eine wertvolle Ressourcennutzung werden. Wer sich auf neue oder veränderte Aufgaben vorbereiten will, muß zwangsläufig Slack ansparen. Die erzielte Wirtschaftlichkeit einer schlanken Organisation steht in keinem Verhältnis zu den schwer quantifizierbaren Verlusten an Flexibilität und Kreativität, der Wahrnehmung von Chancen, der Abwehr krisenhafter Herausforderungen und an Wachstumspotentialen.

In der Regel werden Organisationsstrukturen danach beurteilt, ob die eingesetzten Mittel kostenoptimal ausgelastet sind. Für Nicht-Routineaufgaben ist diese Betrachtungsweise schlecht geeignet. Kritik an Bürokratien wird vor allem von unternehmenspolitischen und humanistischen Theorien geübt. Bürokratien seien inflexibel und wenig anpassungsfähig gegenüber Veränderungen in der Umwelt und gäben dem Organisationsmitglied kaum Gelegenheit zu Persönlichkeitsentfaltung und -entwicklung. Trotz dieser Kritik herrscht bürokratisches Denken vor allem in großen Organisationen.

Neben dem bürokratischen Ansatz zu den drei Konzepten Redundanz, Slack und lose Kopplung, scheint der Informationsverarbeitungsansatz von Galbraith zweckmäßig.[5] Er geht davon aus, daß mit zunehmender Unsicherheit und Komplexität einer Aufgabe die Menge an Informationen steigt, die zwischen Organisationsmitgliedern im Zuge der

Aufgabenbewältigung ausgetauscht werden müssen. Für die Bewältigung solcher Situationen schlägt Staehle zwei unterschiedliche Vorgehensweisen vor:[6]

- *Reduktion des Informationsbedarfs*
  durch Bildung von Organizational Slack und
  Schaffung einer Strukturredundanz

- *Verbesserung der Kommunikation*
  durch Formalisierung und Computerisierung bzw.
  Vernetzung über lose Kopplung

Aus der systemtheoretischen Perspektive vertritt Scott die Auffassung, das Organisationen als offene Systeme prinzipiell zwei zentrale Strategien verfolgen:[7]

- *Pufferstrategien:* Sie verfolgen die Absicht, den technischen Kern gegenüber Umwelteinflüssen zu schützen, um die Sicherheit und Stabilität des Systems zu gewährleisten. Diese Vorgehensweise wird insbesondere beim Kern des Linux-Betriebssystems sichtbar. Dieser ist so organisiert, daß Linus Torvalds die letzte Entscheidung über Neuerungen fällt, die in den Kern des Betriebssystems eingehen. Seine direkten Mitarbeiter filtern die von anderen Entwicklern zugetragenen Neuerungen. Durch diesen Mechanismus wird der technische Kern des Betriebssystems vor Umwelteinflüssen geschützt und die Sicherheit und Stabilität des Systems erhalten.[8]
- *Verknüpfungsstrategien:* Sie verfolgen das Ziel, die peripheren Subsysteme in eine optimale Vernetzung mit den Umweltsystemen zu bringen, um den Ressourcenzugang sicherzustellen. Als Subsystem könnte man den Entwicklerkern einer Community bezeichnen, die Benutzer als „Umweltsystem“. Die Verknüpfungsstrategie besteht nun darin die Kommunikation zwischen Kernentwickler und Benutzer so effizient wie möglich zu gestalten. Dabei werden alle verfügbaren Internet-Dienste, wie E-Mail, Newsgroups, Mailinglisten, usw. eingesetzt.

---

5 Vgl. Galbraith, J. R.: Organization design: An information processing view, in: Interfaces, 3 (1974), S. 28.

6 Vgl. Staehle, W. H.: Redundanz, Slack und lose Kopplung, in: Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 318.

7 Vgl. Scott, W.R.: Organizations: rational, natural, and open systems, 2. Aufl., (Engelwood Cliffs) New Jersey 1987, S. 76.

8 Vgl. Torvalds, L.: Der Pragmatiker der freien Software, in: O’Reilly & Associates, Inc. (Hrsg.): Open Source

Diese nicht-bürokratischen Koordinationsinstrumente sollten bezüglich ihrer Effizienz mit den klassisch-bürokratischen verglichen werden, dabei spielen die Kosten eine wichtige Rolle. Folgende Faktoren sind hier Kostenverursacher:

- *Slack* verursacht Kosten temporär nicht genutzter Ressourcen.
- *Redundanz* verursacht Kosten aus dem Verzicht auf Spezialisierung, Zentralisation und Economies of scale.
- *Computerisierung* verursacht Kosten der Standardisierung und der zusätzlichen Investition in Hard- und Software.
- *lose Kopplung* verursacht Kosten an Arbeitszeit zur Interaktion zwischen Arbeitsgruppen.

Zur ökonomischen Beurteilung der Koordinationsinstrumente sind folgende Fragestellungen von Bedeutung:[9]

1. Stehen den Kosten der nicht-bürokratischen Koordinationsinstrumente Erträge gegenüber, die überproportional größer sind als die Kosten?
2. Werden mit den nicht-bürokratischen Koordinationsinstrumenten Funktionen offener Systeme bereitgestellt, die mit mechanistisch-bürokratischen Konzepten nicht zu bewältigen sind so daß keine kostenoptimalen Alternativen bestehen?

2.1 Redundanz

Obwohl das Redundanz-Konzept aus der Informationstheorie stammt, ist es eng mit dem Slack-Konzept verbunden. In der Informationstheorie ist Redundanz eine Bezeichnung für das Vorhandensein von optionalen Elementen in einer Nachricht, die keine zusätzliche Information liefern, sondern lediglich die beabsichtigte Grundinformation stützen. Der zu Redundanz komplementäre Begriff ist Freiheitsgrad oder Varietät und stellt ein Maß für die Unordnung eines Systems dar. Bei Verwendung eines redundanten Schemas benötigt man beispielweise bei der Übermittlung einer Nachricht mehr Zeit als bei einem nichtredundanten

- kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 33.

9 Vgl. Staehle, W. H.: Redundanz, Slack und lose Kopplung, in: Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 319.

Schema. Dieser Mehraufwand stellt aber insofern keine Verschwendung dar als die Nachrichtenübermittlung in der Realität meist gestört ist und Redundanz erforderlich macht.[10] Der Redundanzbegriff in der Systemtheorie beschäftigt sich mit Redundanz und Varietät von Entscheidungen in Systemen. Die Redundanz in einem System ist demnach hoch, wenn ein Beobachter bei Kenntnis des Entscheidungsprogramms die Entscheidung vorhersagen bzw. das System durchschauen kann. Die Varietät eines Systems ist hoch, wenn die Redundanz gering ist und eine sichere Prognose aufgrund der Vielzahl verschiedenartiger Entscheidungsmöglichkeiten nicht existiert.[11] Normalerweise wird man davon ausgehen, daß Systeme dazu tendieren werden ihre Redundanz zu erhöhen. Es kann aber auch sein, daß ein System auf strukturelle Veränderungen seiner Umwelt mit einer Erhöhung seiner Varietät reagiert.

Nachdem die informationstheoretischen und systemtheoretischen Ansätze die Information in den Vordergrund ihrer Betrachtung der Redundanz stellten, kann noch eine weitere Art von Redundanz unterschieden werden, die sogenannte Strukturredundanz. Sie hat in Organisationen folgende Ausprägungen:[12]

- *Redundanz von Teilen*
  Funktionssicherheit wird durch Austausch, Hinzufügung und Verdopplung von Teilen gewährleistet. Betrachtet man Software-Programme als „Teile“, so findet man bei Open Source Software sehr viele gleich und ähnliche Teile.
- *Redundanz von Funktionen*
  Funktionssicherheit des Systems wird durch Erhöhung der Funktionsredundanz in den Teilen gewährleistet. Funktionen zur Steuerung und Koordination von freien Software-Projekten werden von den Community-Mitgliedern wahrgenommen. Wichtige Funktionen werden stets von mehreren Mitgliedern ausgeführt.
- *Redundanz von Beziehungen zwischen den Teilen*
  Durch offene und lizenzgebührfreie Standards sind Communities in der Lage Software anderer Communities einzusetzen bzw. weiter zu verwenden. So entstehen redundante

[10] Vgl. Staehle, W. H.: Redundanz, Slack und lose Kopplung, in: Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 321-322.
[11] Vgl. Luhmann, N.: Soziologische Aufklärung 2. Aufsätze zur Theorie der Gesellschaft, (Westdeutscher Verlag) Opladen 1982, S. 150-152.
[12] Vgl Beinum, H. van: New Technology and Organizational Choice, in: QWL Fokus 6 (1988), S.4.

Beziehungen zwischen Software-Programmen, Software-Bibliotheken und Datenbanken.

Das organische Organisationsbild übertragen auf Unternehmen begünstig neue Formen der Arbeitsorganisation, wie beispielsweise teilautonome Arbeitsgruppen als auch die parallele Entstehung von Teams, die gleiche oder ähnliche Aufgaben verrichten, um:

- fristgerechte Aufgabenerfüllung sicherzustellen. Open Source Communities haben keine Frist, zu der sie Software fertigstellen müssen. Dies bedeutet nicht, daß sie sich nicht selbst Fristen setzen, um eine kontinuierliche Weiterentwicklung zu gewährleisten.[13] Im Gegensatz zu Unternehmen müssen diese Fristen jedoch nicht zwingend eingehalten werden und im allgemeinen wird es einer Community positiv ausgelegt, wenn sie die Frist zugunsten der Qualität und Stabilität verlängert.[14]
- aufgrund von Wettbewerb zwischen Teams qualitativ höherwertige Ergebnisse als bei Einfachbesetzung zu erreichen. Open Source Communities organisieren sich grundsätzlich kooperativ, d. h. es gibt keine Einfachbesetzung, jedes Mitglied ist in seiner Funktion austauschbar. Wettbewerb zwischen Entwicklerteams findet bei großen Projekten innerhalb der Community statt, darüber hinaus auch außerhalb, wie beispielsweise die Projekte GNOME und KDE, die einen Teil ihrer Entwicklungsdynamik und Qualität dem Vergleich der Benutzer und damit dem Wettbewerb zu verdanken haben.
- Überlastung bei Einfachbesetzung zu reduzieren. Software-Projekte der Größe einer Community sind meist zu komplex und aufwendig als daß ein einzelner Entwickler diesen Arbeitsaufwand alleine bestreiten könnte.
- Fehler als Voraussetzung für individuelles und organisatorisches Lernen zu ermöglichen ohne den Routineablauf zu gefährden.
- Selbstorganisation im Sinne von teilautonomen Arbeitsgruppen zu ermöglichen.[15] Die Aussicht für Community-Mitglieder auf eine Zusammenarbeit mit erfahrenen Software-Experten motiviert zur Selbstorganisation.[16]

---

13 Vgl. Dahlheimer, K.: Freiberufler-Info, <URL: http://www.computerwoche.de/info-point/pop-news/drucken.cfm?id=12769>, online: 09.11.99.

14 Losi, S.: Stephanie Chats with Uwe Thiem, <URL: http://go.borders.com/features/slo99092.xcv >, online: 21.12.99.

15 Vgl. Scholz, C.: Strategische Organisation: Prinzipien zur Vitalisierung und Virtualisierung, (Moderne Industrie) Landsberg/Lech 1997, S. 189.

16 Vgl. Dahlheimer, K.: Freiberufler-Info, <URL: http://www.computerwoche.de/info-point/pop-news/drucken.cfm?id=12769>, online: 09.11.99.

Warum sollte es nicht auch Wettbewerb in Organisationen geben? Wettbewerb und Redundanz sind nicht identisch. Alle Wettbewerbssysteme sind zwar redundant, aber nicht jede Strukturredundanz ist kompetitiv. Dies bedeutet, daß sich redundante Strukturen entwickeln können ohne zunächst im Wettbewerb zueinander zu stehen. Die organisatorische Praxis zeigt jedoch, daß geplant oder ungeplant entstandene Parallelstrukturen in Konkurrenz zueinander um knappe Ressourcen treten. Dieser Zustand ist auch bei der Open Source Software-Entwicklung zu beobachten. Die ersten drei Entwicklungsphasen der Communities haben redundante Strukturen entwickelt und die Software-Projekte waren nicht sehr vom Wettbewerb geprägt. Mit der Phase des Community-Pluralismus sind die ausgebauten redundanten Strukturen aufgrund ihrer Vielzahl teilweise in Wettbewerb getreten. Systematisch geförderter Wettbewerb um knappe Ressourcen ist ein Mittel, die nicht effiziente Nutzung von Ressourcen zu reduzieren. Unternehmen, wie Red Hat als Linux-Distributor binden Kernentwickler aus Communities durch Verträge an sich und fördern damit die jeweilige Community und den Wettbewerb zwischen Open Source Projekten.

Strukturredundanz verursacht zunächst Mehrkosten. Worin liegt aber der potentielle Mehrertrag? Für die serielle Kopplung gilt, daß die Kette nur so stark ist wie ihr schwächstes Glied und die Zuverlässigkeit des Serien-Systems höchstens gleich der Zuverlässigkeit des unzuverlässigsten Elements ist, d. h. ein Serien-System kann nie besser funktionieren als sein schwächstes Teil. Seine Zuverlässigkeit ist somit begrenzt. Für redundante Systeme gilt, daß durch eine Duplikation des Systems dieses mindestens so stark oder stärker wird als sein stärkstes Teil. Die Zuverlässigkeit eines Parallelsystems ist immer größer oder mindesten gleich der Zuverlässigkeit seines zuverlässigsten Elements, d. h. durch parallele Anordnung beliebig vieler Elemente läßt sich eine beliebig hohe Zuverlässigkeit erreichen.[17] Das Open Source Software-Entwicklungsmodell nach Raymond besagt, daß Debugging, das Austesten von Software und Fehlerbeseitigen, parallelisierbar ist. Durch die Einbeziehung der Benutzer als Mitentwickler wird Software schnell verbessert und die Effizienz der Fehlersuche gesteigert. Ist die Basis an Mitentwickler groß genug, dann wird nahezu jedes Problem schnell charakterisiert und eine Lösung gefunden werden. Raymond kommt zu der Erkenntnis[18], daß

[17] Vgl. Zeep, W.: Redundanz - Ein Mittel zur Steigerung der Zuverlässigkeit von technischen Systemen, in: Bussmann, K. F.; Mertens, P. (Hrsg.): Operations Research und Datenverarbeitung bei der Instandhaltungsplanung, Stuttgart 1968, S. 89.

[18] Vgl. Raymond, E. (Hrsg.): The Cathedral and the Bazaar,

eine Basis an guten und weniger guten Mitentwicklern eine schnellere Entwicklung von Software ermöglicht, als dies einer Gruppe von sehr guten Entwicklern möglich ist. Die Effizienz in parallelen Systemen ist bei der Software-Entwicklung offensichtlich größer als in seriellen Systemen. Das Konzept der seriellen Anordnung geht von der Annahme aus, daß sich die Sicherheit von Systemen durch die Perfektionierung seiner Teile erhöhen läßt. Mit einer paralleler Anordnung ist es aber auch möglich ein zuverlässiges System zu gestalten, daß aus tendenziell unzuverlässigen Teilen besteht.[19]

2.2 Lose Kopplung

Mehrertragspotentiale redundanter Strukturen lassen sich um so besser realisieren, desto unabhängiger die parallelen Teams arbeiten. Je autonomer die Organisationseinheiten agieren können, desto effizienter sind redundante Strukturen. Daher das dritte Konzept neben Redundanz und Slack, das Konzept der losen Kopplung von Organisationseinheiten, besonders im Sinne von Subsystemen. Im Gegensatz zu automatischen Kontrollsystemen, die enge Kopplungen aufweisen, damit das System auf Input bzw. Störgrößen schnell reagieren kann, weisen lose Kopplungen lokal stabile Subsysteme auf, die Störungen an anderen Stellen nicht beeinflussen. Enge Kopplung bedeutet Inflexibilität, hohe Anfälligkeit gegenüber nicht prognostizierten Störungen und zeitliche Abhängigkeit der vor- und nachgelagerten Prozesse. Das Funktionieren einer losen Kopplung von Organisationseinheiten erfordert andere organisatorische und personelle Voraussetzungen als enge Kopplung. So beispielsweise eine hohe Qualifikation der Mitarbeiter, permanente Weiterbildung, Vertrauensorganisation, autonome, dezentrale Organisationseinheiten, also alle Voraussetzungen der Selbstorganisation. Eine Vertrauensorganisation impliziert ein Verzicht auf direkte Kontrolle und ein Netz informaler Kommunikationsbeziehungen. Selbstorganisation setzt permanentes individuelles und organisationales Lernen voraus[20] und erfordert in zunehmenden Maße eine Erhöhung organisatorischer Redundanz sowie organizational Slack, so daß viele sich selbst koordinierende Individuen oder Gruppen entstehen, und mit „überflüssigen“ Funktionen

<URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>, online: 23.09.99.

19 Vgl. Chisholm, D.: Coordination without hierarchy-Informal structures in multiorganizational systems, Berkeley, Los Angeles, London 1989, S. 183.

20 Vgl. Scholz, C.: Strategische Organisation - Prinzipien zur Vitalisierung und Virtualisierung, (Moderne Industrie) Landsberg/Lech 1997, S. 189-194.

ausgestattet möglichst viele Funktionen wahrnehmen können.[21] Branchen, die auf schwankende und individuelle Kundenwünsche reagieren müssen, haben lose gekoppelte Produktionssysteme, typischerweise kommt Gruppenarbeit in sogenannten Fertigungsinseln zum Einsatz. Weick[22] empfiehlt relativ kleine in sich fest gekoppelte Einheiten (stabile kollektive Kleingruppen), in denen kurze, aber häufige Interaktionen stattfinden. Diese sollen nur locker (lose Kopplung) mit anderen organisatorischen Subsystemen verbunden werden. Die Kern-Entwickler der Communities sind kleine und fest gekoppelte Einheiten, die relativ stabil sind und häufig kommunizieren. Andere, nicht zur Kern-Gruppe gehörende Mitglieder der Community, sind eher lose miteinander gekoppelt. Granovetter berichtet, daß lose Kopplungen besser geeignet sind, um Netzwerke von Personen miteinander zu verknüpfen. [23] Es scheint auf den ersten Blick nicht plausibel, daß lose Kopplungen sich besser zur Verbindung von kleinen Gruppen untereinander eignen und die kleinen Gruppen selbst am besten durch eine enge Kopplung zusammengehalten werden. Ein weiterer Vorteil loser Kopplungen wird von Staehle darin gesehen, daß Störungen in kleinen Organisationseinheiten zunächst auf diese begrenzt bleiben und nicht wie bei enger Kopplung auch schnell auf das Gesamtsystem übergreifen. Lose Kopplungen erleichtern auch die lokale Anpassung von Subsystemen an Umweltveränderung bzw. neuen Anforderungen.

2.3 Anwendung der Konzepte Slack, Redundanz und lose Kopplung

Zunächst wird hier in drei Organisationsformen unterschieden, bevor diese gegenübergestellt werden. Die *hierarchische Organisation* und die *Parallel-Organisation*, welche Funktions- und Strukturredundanz enthält. Die Mitglieder der Parallelorganisation sind zugleich Mitglieder der Linienorganisation, d. h. sie arbeiten nur temporär in der Parallelorganisation. Diese Organisationsformen werden in der folgenden Tabelle der *Open Source-Organisation* gegenübergestellt.

Empfehlungen und Entscheidungsmodelle für Open Source-Organisationen können, je nach Entscheidungssituation, mehr zum einem oder anderen Extremtyp tendieren. In der

[21] Vgl. Probst, G.: Selbstorganisation und Entwicklung, in: DU 41 (1987) 1987, S. 242.255.
[22] Vgl. Weick, K. E.: The Social Psychology of Organizing, 2. Aufl. (Addison Wesley) London 1979, S. 110.
[23] Vgl. Granovetter, M. S.: The strength of weak ties, in: AJS 78 (1973) S. 1376.

ökonomischen Tradition wird man dann nach einem optimalen Slackniveau, also einem optimalen Grad an Redundanz oder Kopplung/Vernetzung suchen. Die Quantifizierung von verhaltenswissenschaftlichen Variablen wird aber sehr wahrscheinlich scheitern. Sinnvoller erscheint es, betriebliche Situationen zu beschreiben oder nachträglich (ex post) zu beschreiben, wie Open Source sich im Zeitablauf entwickelt hat und dann aus diesen Erkenntnissen Gestaltungsempfehlungen zu konstruieren, die zu bestimmten Szenarien führen. Abschließend läßt sich anhand der Tabelle verdeutlichen, daß die Open Source-Organisation in vielen Situationen hinsichtlich ihrer Effizienz der hierarchischen als auch der parallelen Organisation überlegen sein kann.

Hierarchische Organisation	Parallel-Organisation	Open Source-Organisation
Routine, geringe Unsicherheit	Problemlösung, hohe Sicherheit	störende Unzulänglichkeit
Ziel ist „Produktion“	Ziel ist „Organisation“	Ziel ist Problemlösung/-beseitigung
feste Stellenbeschreibung	flexible, rotierende Aufgabenzuweisung	persönlich induzierte, volontäre Zurverfügungstellung
Qualifikation vor der Aufgabenübernahme	Qualifikation während der Aufgabenübernahme	Qualifikation vor und während der Aufgabenübernahme
langer Dienstweg	kurzer Dienstweg	kein Dienstweg
Zielbildung top-down	Zielbildung auch bottom-up	Zielbildung best-wins
Anreize: Bezahlung	Anreize: Lernchancen, soziale Kontakte, Anerkennung	Anreize: Studium, soziale Kontakte, Anerkennung, Bezahlung (aber unabhängig)
funktionale Spezialisierung	Diagonale Verknüpfungen	fachliche Spezialisierung, teilweise auch funktionale und diagonale
Amtsautorität	Personale Autorität	Fachliche Autorität, die im Extrem auch personal wird

Tab. 5: Organisationsformen im Vergleich

3 Open Source – Lizenzen

Nachfolgend werden die wesentlichen Lizenzmodelle vorgestellt und unter dem Aspekt des

Open Source Konzeptes diskutiert. Hieran schließt sich eine Gegenüberstellung der Lizenzmodelle an.

3.1 Public Domain

Public Domain (PD) ist eine US-amerikanische Besonderheit, die nach deutschen Urheberrecht nicht zulässig ist. In den USA stehen alle durch staatliche Unterstützung entstandene Software-Ergebnisse an Hochschulen, Forschungseinrichtungen etc. jedem US-Bürger uneingeschränkt zur Verfügung. Autoren der Software geben dabei alle Urheberrechte auf.[24] Public Domain Software hat kein Copyright. Sie ist freie Software und es ist jedem erlaubt proprietäre Versionen zu erstellen.

3.2 Shareware

Shareware ist eine besondere Art des Vertriebs von Software.[25] Ziel ist es, diese für jeden verfügbar zu machen. Shareware-Programme werden ausschließlich in Binärform von Autoren an die Öffentlichkeit weitergegeben. Der Autor verzichtet nach dem Urheberrecht auf das Recht, die Verbreitung einzuschränken. Daraus ergibt sich für die Anwender die Möglichkeit, jederzeit und auf jede Art in den Besitz einer Kopie eines Shareware-Programms zu kommen. Dies kann per Diskette, CD-ROM, das Internet oder andere Netze geschehen. Der Autor nimmt darauf keinen Einfluß. Shareware Produkte sind voll funktionstüchtige und frei verteilbare Programme, haben aber eine Lizenz, die private und kommerzielle Anwender schließlich erwerben sollen. Viele Programme, wie beispielsweise WinZip, nutzen die Vorteile der Sharewareverteilung.[26]

Die Kosten, die bei der Verteilung entstehen, werden zwischen dem Autor, den Anwendern und eventuell zwischengeschalteten Dienstleistern aufgeteilt. Es hat sich dabei folgendes Geschäftsmodell etabliert:

---

[24] Vgl. OASE (Hrsg.): OASE: Was ist Shareware, <URL: http://members.aol.com/wwwoase/oase/defshw.htm>, online: 18.11.99.

[25] Vgl. OASE (Hrsg.): OASE: Was ist Shareware, <URL: http://members.aol.com/wwwoase/oase/defshw.htm>, online: 18.11.99.

[26] Vgl. OpenSource.Org (Hrsg.): Halloween I: Open Source Software -- A (New?) Development Methodology, <URL: http://www.opensource.org/halloween/halloween1.html>, online: 12.11.99.

- Der Autor übernimmt die Kosten, die bei der Übermittlung der Shareware-Kopie an geeignete Stellen entstehen. Dies sind in der Hauptsache Verbindungsentgelte, die zum Beispiel beim Hochladen auf einen FTP-Server entstehen.
- Der Anwender kommt für die Kosten auf, die zur Erlangung einer Shareware-Kopie anfallen. Auch hier sind das im wesentlichen Verbindungsentgelte für das Runterladen von einem Server.
- Schalten sich zwischen Autor und Anwender Dritte ein, die für ihre Dienstleistung ein Entgelt erheben, muß dies vom Autor genehmigt werden. Oft werden viele verschiedene Shareware-Programme auf CD-ROM systematisch zusammengefaßt und dann verkauft oder zum Beispiel Zeitschriften beigelegt.

Die ursprüngliche Idee der (kosten)freien Kopie hat sich nur bedingt durchgesetzt. In der Vergangenheit haben viele Händler die Gelegenheit des nahezu rechtsfreien Raums ausgenutzt und sich ihre Dienstleistung teuer bezahlen lassen. Mit der entsprechenden Rechtssprechung gemäß Urheberrechtsergänzung und der zunehmenden Verbreitung des Internets ist dieses Problem aber entschärft worden. Wenn man die Entwicklung der Übertragungskosten (Telefonnetz, Providergebühren) betrachtet, dann wird in Zukunft immer weniger für die Erlangung einer Shareware-Kopie zu zahlen sein. Ein Trend, der für Autoren und Anwender gleichermaßen von Vorteil ist.

Der Shareware-Autor gibt die Nutzungsrechte für einen bestimmten Zeitraum frei. Was dies konkret bedeutet, was wirklich mit dem Programm erlaubt ist, legt der Autor fest. Gemäß der Rechtsprechung ist alles nicht erlaubt, was nicht explizit festgelegt ist. Der Zeitraum wird auch als Testzeitraum bezeichnet und liegt üblicherweise zwischen 10 und 60 Tagen. Dabei werden drei Möglichkeiten der Zeitmessung für den Testzeitraum unterschieden:

1. *Vergangene Tage ab Installationsdatum:* Hierbei ist der Anwender im Nachteil, der das Programm nicht täglich nutzt.

2. *Wirklich genutzte Testtage:* Bei der Zählung der wirklichen Testtage spielt das Installationsdatum und die Anzahl der Programmaufrufe keine Rolle. Dem Anwender stehen echte Testtage zur Verfügung.

3. *Anzahl der Programmaufrufe:* Eine Sonderform stellt die Zählung der Programmaufrufe dar. Es ist für Autoren und Anwender schwer abzuschätzen, wieviel Aufrufe für einen ausreichenden Test nötig sind. Deshalb ist diese Verfahren in der Praxis nur bedingt tauglich.

Nach Ablauf des Testzeitraums ist eine weitere Nutzung ohne die Entrichtung einer vom Autor festgelegten Lizenzgebühr nicht erlaubt. Wird die Software ohne Entrichtung der Lizenzgebühr weiter eingesetzt, so wird gegen das Urheberrecht verstoßen. Das Shareware-Konzept wird von seinen Vertretern als anwenderfreundlich bezeichnet. Sie appellieren an das gute Gewissen der Anwender und mahnen, die gewährten Rechte nicht über das erlaubte Maß hinaus zu strapazieren. Sie argumentieren, daß Autoren für ihre Arbeit entlohnt werden wollen und wenn dies nicht (ausreichend) geschehe, würden sie ihre Dienste nicht mehr zur Verfügung stellen (können).

3.3 Freeware

Freeware ist das lizenzkostenfreie Pendant zur Shareware mit den gleichen Regeln bezüglich des Vertriebs und der Weitergabe. Der Testzeitraum entfällt. Der Autor kann allerdings die Nutzung auf einen bestimmten Nutzerkreis einschränken, der keine Lizenzgebühren zu entrichten hat. Verstößt ein Anwender gegen diese Lizenzbedingungen, kann der Freeware-Autor dagegen rechtliche Schritte einleiten.[27] Freeware-Programme werden ebenfalls ausschließlich in Binärform veröffentlicht. Ein Beispiel ist der Internet Explorer von Microsoft.[28]

3.4 MIT License

Die MIT Lizenz erlaubt jeder Person eine kostenlose Kopie der Software und dazugehöriger Dokumentation zu erhalten sowie ohne Einschränkung mit der Software zu handeln. Weiterhin gestattet sie die Rechte die Software ohne Einschränkung zu benutzen, zu kopieren, zu modifizieren, zu vermischen, zu veröffentlichen, zu verteilen, unterzulizenzieren und/oder

[27] Vgl. OASE (Hrsg.): OASE: Was ist Shareware, <URL: http://members.aol.com/wwwoase/oase/defshw.htm>, online: 18.11.99.

[28] Vgl. OpenSource.Org (Hrsg.): Halloween I: Open Source Software -- A (New?) Development Methodology,

Kopien der Software zu verkaufen. Diese Lizenzbedingungen gelten auch für alle Kopien und Portierungen der Software.

3.5 BSD License

Die BSD Lizenz der University of California, Berkeley entspricht seit dem 22. Juli 1999 der oben beschriebenen MIT Lizenz bis auf eine Ausnahme: Weder der Name der Copyrighthalter noch die Namen der zur Software oder Dokumentation Beitragenden dürfen ohne besondere schriftliche Erlaubnis benutzt werden, um Produkte zu bewerben, die von der Software abgeleitet sind.

Open Source Software, die unter der BSD-Lizenz steht, stammt von einem geschlossenen Team von Entwicklern. Sowohl die Binaries als auch der Quellcode darf lizenzkostenfrei von jedem genutzt und verteilt werden. Obwohl Benutzer den Quellcode auch modifizieren dürfen, werden von den Entwicklern in der Praxis selten Modifikationen in ihre Software übernommen.[29]

3.6 Artistic License

Die Artistic License möchte dem Autor bzw. dem Copyrighthalter einer Sache (in diesem Falle Software) eine Art „künstlerische Kontrolle“ über sein Werk und den Entwicklungsprozeß sichern, wobei Dritte das Recht haben, das Werk zu nutzen und in einer üblichen Art und Weise weiterzuverteilen. Außerdem haben sie das Recht das Werk zu modifizieren. Weiterhin gelten für die Artistic License folgende Regeln:

- Kopien der Standard Version können ohne Beschränkung weiterverteilt werden, sie müssen nur die ursprünglichen Copyrightbestimmungen und -bemerkungen etc. enthalten.
- Der Name des Copyrighthalters darf ohne besondere schriftliche Erlaubnis nicht dazu benutzt werden, Produkte zu bewerben, die von dieser Software abgeleitet sind.
- Werden Fehler beseitigt, das Paket portiert oder modifiziert und kommen diese

<URL: http://www.opensource.org/halloween/halloween1.html>, online: 12.11.99.

[29] Vgl. OpenSource.Org (Hrsg.): Halloween I: Open Source Software -- A (New?) Development Methodology, <URL: http://www.opensource.org/halloween/halloween1.html>, online: 12.11.99.

Änderungen vom Copyrighthalter, kann dieses veränderte Paket wieder als Standard Version bezeichnet werden.

- Wird das Paket von anderen als dem Copyrighthalter verändert, muß in jedem veränderten Dokument bzw. in jeder veränderten Datei eine Dokumentation über Art und Datum der Änderung zu finden sein.
- Das Paket kann auch in Maschinencode bzw. als ausführbares Programm (Binary) zur Verfügung gestellt werden, wenn einer der folgenden Punkte eingehalten wird:
  - Es ist erlaubt eine „vernünftige“ Kopiergebühr für die Distribution zu verlangen. Für unterstützende Dienstleistungen der Distribution kann jede Gebühr verlangt werden. Für die Distribution selbst darf keine Gebühr verlangt werden. Ist das Paket Teil einer größeren Distribution, darf der Distributor das Paket nicht als sein eigenes bewerben.
  - Skripte und Softwarebibliotheken, die Eingaben oder Ausgaben für das eigentliche Programm produzieren, fallen nicht automatisch unter das Copyright des Paketes. Sie gehören demjenigen, der sie generiert und können kommerziell sein als auch dem Paket beigelegt werden.
  - C oder Perl Unterprogramme, die dem Paket beigelegt werden, sind kein Bestandteil dieses Paketes.

3.7 MPL und QPL

MPL steht für Mozilla Public License. Es handelt sich dabei um die Lizenz, unter der Netscapes freier Open Source Browser Mozilla steht.[30] Durch die Preispolitik des Mitbewerbers Microsoft ist die Annäherung Netscapes an die Open Source Welt eingeleitet worden. Frank Hecker von Netscape schrieb den durch Eric Raymonds „The Cathedral and the Basar“[31] beeinflußten Aufsatz „Setting Up Shop: The Business of Open Source“[32], der die Unternehmensleitung von Netscape am 22.09.1998 dazu veranlaßte, die Freigabe des Browsercodes bekanntzugeben.[33]

---

30 Vgl. The Mozilla Organisation (Hrsg.): Mozilla.org, <URL: http:www.mozilla.org/>, online: 08.12.99.

31 Vgl. Raymond, E. (Hrsg.): The Cathedral and the Bazaar, <URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>, online: 23.09.99.

32 Vgl. Hecker, F.:Setting Up Shop: The Business of Open-Source Software, <URL: http://people.netscape.com/hecker/setting-up-shop.html>, online: 11.12.99.

33 Vgl. Netscape Inc. (Hrsg.): Press Release, <URL: http://www.netscape.com/newsref/pr/newsrelease558.html>, online: 25.11.99.

QPL steht für Qt Public License der Firma Troll Tech. Unter dieser Lizenz steht die freie Version der Software Bibliothek Qt, die durch den Einsatz im KDE Projekt bekannt wurde.[34] Anfangs durfte die Software Bibliothek Qt zwar für das Betriebssystem Linux frei benutzt werden, aber mit zunehmender Verbreitung von KDE regte sich gegen die ansonsten kommerziellen Lizenzbestimmungen von Qt ein starker Widerstand in der Community. Vor allem, weil nicht garantiert war, daß Qt auf lange Sicht frei bleiben würde und damit auch die nicht unerheblichen Modifikationen, die von der Community zur Weiterentwicklung von Qt beitgetragen wurden. Als der Druck auf Troll Tech zu groß wurde, hat das Unternehmen seinerzeit mit der QPL die langfristig freie Verfügbarkeit von Qt sichergestellt, auch über das Ende von Troll Tech hinaus. Dies wurde von der Community positiv honoriert[35] und verhalf KDE zu weiterem Wachstum, da zukünftig kein Distributor die Auslieferung von KDE mit Verweis auf Qt verweigern kann.

Den Lizenzen MPL und QPL ist gemein, daß sie von Unternehmen stammen, die ein bestimmtes bereits am Markt existierendes Produkt mit der Open Source Welt verbinden wollten. Kommerzielle Software beinhaltet oft auch kommerzielle Software von Dritten und benutzt fremde Patente, für die Unternehmen bezahlen müssen. Da eine Überführung solch großer Software Pakete in die Open Source Welt schwierig ist und nicht alle Open Source Lizenzen dafür geeignet sind, wurden die MPL und QPL Open Source Lizenzen entwickelt.

3.8 Apache

Die Apache Group stellt eine sehr einfache Lizenz[36] zur Verfügung, die die Distribution von Quellcode und Maschinencode mit oder ohne Modifikationen erlaubt, wenn folgende Bedingungen zutreffen:

- Distributionen des Quellcodes müssen das Copyright der Apache Group, diese Liste der Bedingungen und den Haftungsausschluß enthalten.
- Distributionen des Maschinencodes muß das Copyright der Apache Group, diese Liste der

[34] Vgl. Troll Tech (Hrsg.): The QPL, an Open Source License, <URL: http://www.troll.no/qpl/>, online: 12.12.99.
[35] Vgl. Troll Tech (Hrsg.): Announcement: Open Source, <URL: http://www.troll.no/announce/qpl.html>, online: 12.12.99.
[36] Vgl. The Apache Software Foundation (Hrsg.): Apache Group License,

Bedingungen und den Haftungsausschluß in ihrer Dokumentation oder auf eine andere Art und Weise beigelegt werden.

- Wird Apache Group Software und ihre Merkmale beworben, dann muß folgende Bestätigung sichtbar sein: „This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/)."
- Weder „Apache Server“ noch „Apache Group“ dürfen ohne besondere schriftliche Erlaubnis benutzt werden, um Produkte zu bewerben, die von der Software abgeleitet sind.
- Abgeleitete Software darf weder „Apache“ genannt werden noch darf „Apache“ in ihren Namen stehen ohne besondere schriftliche Erlaubnis der Apache Group.
- Jede Distribution, in welcher Form (Quell- oder Maschinencode) auch immer, muß folgende Bestätitgung enthalten: „This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/).“

Die Software der Apache Group besteht aus freiwilligen Beiträgen von sehr vielen Individuen zugunsten der Apache Group und basiert ursprünglich auf Public Domain Software (Software zum öffentlichen Gebrauch und ohne jegliche Schutz), die am National Center for Supercomputing Applications der University of Illinois, Urbana-Champaign entwickelt und geschrieben wurde. Aufgrund dieser Herkunft ist auch ein kommerzieller Einsatz der Software ohne Einschränkungen bezüglich der Distributionsbedingungen (Verkaufspreis, Quellcode etc.) möglich. Dies unterscheidet die Apache Lizenz von anderen Open Source Lizenzen, welche zwingend, also nicht fakultativ die kommerziellen Distributionen mit Auflagen belegen.

3.9 GPL

Die am häufigsten angewandte Open Source Software Lizenz ist die GNU General Public License, kurz GPL[37]. Sie ist einer der ältesten, umfangreichsten und weitestgehenden Lizenzen überhaupt. Richard Stallman gehört zu den geistigen Vätern und ist ihr unnachgiebigste Verfechter. Die FSF entwickelte das Lizenzverfahren Copyleft, demzufolge es nicht nur

<URL: http://www.apache.org/license.txt>, online: 14.10.99.

[37] Vgl. Free Software Foundation (Hrsg.): GNU General Public License, <URL: http://www.fsf.org/copyleft/gpl.html>, online 16.10.99.

illegal ist, den Quellcode von GNU-Software zurückzuhalten, sondern es auch untersagt ist, den Quellcode von Weiterentwicklungen von GNU-Software geheimzuhalten. Das Dokument, daß dieses Lizenzverfahren beschreibt, ist die GPL.

In der Präambel der GPL sind auch die Formulierungen zu finden, die zum negativen Image[38] sogenannter freier Software beigetragen haben. Dort ist zu lesen, daß die meisten Software Lizenzen nur zu dem Zweck entwickelt wurden, um den Entwickler und Anwendern die Freiheit zu nehmen, Software zu verändern und gemeinsam zu teilen. Mit anderen Worten könnte man sagen, daß Software im Sinne der GPL ein Allgemeingut ist, ohne Eigentümer und für alle verfügbar. Die Bezeichnung „freie Software“ bezieht sich in der GPL auf die Freiheit und nicht auf den Preis.[39] Freiheit bedeutet in diesem Zusammenhang:

- Kopien freier Software verteilen zu dürfen (auch gegen Gebühr),
- Den Quellcode auf Wunsch zu erhalten,
- Die Software zu verändern,
- Teile der Software in neuer freier Software zu benutzen.

Diese Formulierungen wirken auf viele ideologisch und zum Teil sogar sozialistisch, weshalb die Geschäftswelt lange Zeit von der GPL Abstand nahm. Das die GPL eine aus der Sicht der Software Entwickler sehr günstige Lizenz ist, zeigt die Verbreitung der GPL unter freier Software wie beispielsweise Linux.

GPL-basierte Software geht einen entscheidenden Schritt weiter als andere Lizenzen. Während BSD und Apache es erlauben, die Codebasis aufzuspalten und die eigenen Modifikationen unter einer eigenen Lizenz zu stellen, um diese beispielsweise kommerziell zu nutzen, verlangt die GPL, daß alle Codeableitungen wiederum unter der GPL stehen müssen. Die Klausel, abgeleiteten Quellcode ebenfalls unter die GPL stellen zu müssen, war und ist heute noch der umstrittenste Teil der Copyleft-Lizenzierung, aber möglicherweise der

[38] Vgl. OpenSource.Org (Hrsg.): Why `Free Software' Is Too Ambiguous <URL: http://www.opensource,org/free-notfree.html>, online: 12.11.99.

[39] Vgl. Free Software Foundation (Hrsg.): GNU General Public License, <URL: http://www.fsf.org/copyleft/gpl.html>, online 16.10.99.

erfolgreichste.[40]

3.10 LGPL

Die GNU Lesser General Public License (LGPL) der FSF ist der Nachfolger und die Weiterentwicklung der GNU Library Public License.[41] Steht eine Software Bibliothek unter der GPL und wird Software mit dieser Bibliothek verbunden (verlinked), dann muß die Software ebenfalls unter der GPL stehen. Dies verhindert, daß kommerzielle Software aus freier Software entsteht, andererseits wird diese Bibliothek nur für freie Software benutzt werden, was ihre Verbreitung einschränkt. Deshalb gibt es die LGPL, die es bestimmten Bibliotheken erlaubt, mit kommerzieller Software verbunden zu werden. Sie wird deshalb auch „Lesser GPL“ genannt, was zum Ausdruck bringen soll, daß die Freiheit weniger stark als mit der GPL geschützt wird. Trotzdem hat die LGPL unter bestimmten Umständen Vorteile gegenüber der GPL. Diese sind im wesentlichen folgende 3 Situationen:

- Es kann einen speziellen Bedarf an einer möglichst weiten Verbreitung und Nutzung einer bestimmten Software Bibliothek geben, die zum de facto Standard werden soll. Um dies zu erreichen muß es kommerzieller Software erlaubt sein die Bibliothek zu benutzen.
- Häufiger sind freie Software Bibliotheken in Funktionsumfang und Qualität mindestens genauso gut sind wie kommerzielle. In diesen Fällen wäre für die freie Bibliothek nicht viel zu gewinnen, wenn sie nur für freie Software zur Verfügung stünde. Deshalb wird in diesen Fällen gerne die LGPL eingesetzt, um mit kommerziellen Bibliotheken zu konkurrieren.
- In vielen Fällen erlaubt der Einsatz bestimmter freier Software Bibliotheken in kommerzieller Software einer größeren Zielgruppe auch den Einsatz von weiterer freier Software, wie beispielsweise freie Software Entwicklungswerkzeuge und das freie Betriebssystem Linux.

Die LGPL dient hauptsächlich der Verbreitung freier Software und soll den Entwicklern kommerzieller Software die Qualität freier Software demonstrieren. Der Übergang vom

---

[40] Vgl. OpenSource.Org (Hrsg.): Halloween I: Open Source Software -- A (New?) Development Methodology, <URL: http://www.opensource.org/halloween/halloween1.html>, online: 12.11.99.

[41] Vgl. Free Software Foundation (Hrsg.): GNU Lesser General Public License,

Software produzierenden Unternehmen hin zum Software Dienstleiter wird durch die LGPL erleichtert und unter Umständen erst ermöglicht.

3.11 Lizenzvergleich

Die verschiedenartigen Softwarelizenzen werden im folgenden nach ausgewählten Kriterien verglichen. Folgende Tabellen veranschaulichen die einzelnen Lizenz-Bedingungen. Hier wird deutlich, daß jede Lizenz vor einem individuellen Hintergrund und für verschiedene Zielsetzungen erstellt wurde. Die Lizenzen haben dazu geführt, daß sich Open Source Software schnell, mit hoher Qualität und finanziell relativ unabhängig entwickeln konnte.

Software-Lizenz	Frei verfügbarer Quellcode	Erlaubte Modifikationen des Quellcodes	Erlaubte Weiter-verteilung des Quellcodes	Erlaubte Verbindung mit proprietärer Software
PD	x	x	x	x
Shareware	nur ausführbare Programme	x	(x)	
Freeware	nur ausführbare Programme	x	(x)	
MIT-License	(x)	x	x	x
BSD License	(x)	x	x	x
Artistic License	x	x	x	
MPL/QPL	x	x	x	x
Apache	x	x	x	x
GPL	x	x	x	
LGPL	x	x	x	x

Tab. 1: Lizenzbedingungen I

<URL: http://www.gnu.org/copyleft/lesser.html>, online: 14.11.99.

Software-Lizenz	Modifikationen dürfen „privat“ bleiben	Erlaubte Unter-lizensierung	Unentgeltliche Lizenz	Eingeschränkte Nutzung
PD	x	x	x	
Shareware			(x)	x
Freeware			x	(x)
MIT-License	x		x	
BSD License	x		x	
Artistic License	x		x	
MPL/QPL	x		(x)	(x)
Apache	x		x	
GPL			x	
LGPL			x	

Tab. 2: Lizenzbedingungen II

Literatur

Behlendorf, B.: Wir müssen ein Geschäftsproblem lösen, in: O'Reilly & Associates, Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 44-45.

Beinum, H. van: New Technology and Organizational Choice, in: QWL Fokus 6 (1988), S.3-10.

Bleicher, K.; Gomez, P. (Hrsg.): Zukunftsperspektiven der Organisation, Bern 1990.

Bosetzky, H.; Heinrich, P.: Mensch und Organisation, 5. Aufl., (Dt. Gemeindeverlag) Köln 1994.

Bühner, R.: Economies of Speed: Beschleunigung der Abläufe in Unternehmen zur Erhöhung der Wettbewerbsfähigkeit, in: Bleicher, K.; Gomez, P. (Hrsg.): Zukunftsperspektiven der Organisation, Bern 1990, S. 29-43.

Bussmann, K. F.; Mertens, P. (Hrsg.): Operations Research und Datenverarbeitung bei der Instandhaltungsplanung, Stuttgart 1968.

Chisholm, D.: Coordination without hierarchy-Informal structures in multiorganizational systems, Berkeley, Los Angeles, London 1989.

COMPUTERWOCHE (Hrsg.): IBM bündelt Apache und Websphere, <URL: http://www.computerwoche.de/info-point/top-news/drucken.cfm?id=3927>, online: 22.06.98.

CPAN (Hrsg.): Comprehensive Perl Archive Network, <URL: http://cpan.perl.org/>, online: 19.12.99.

Dahlheimer, K.: Freiberufler-Info, <URL: http://www.computerwoche.de/info-point/pop-news/drucken.cfm?id=12769>, online: 09.11.99.

DiBona, C.; Ockmann, S.; Stone, M. (Hrsg.): Open Sources: Voices from the Open Source Revolution, (O'Reilly) Sebastopol 1999.

Diedrich, O.: Umstrittener Vergleich zwischen NT und Linux, <URL: http://www.heise.de/newsticker/data/odi-14.04.99-001/default.shtml>, online: 14.04.99.

Farrell, J.; Shapiro, C.: Dynamic Competition with Switching Costs, in: Rand Journal of Economics 19 (1988), S. 123-132.

Foresight Institute (Hrsg.): Homepage, <URL: http://www.foresight.org/homepage.html>, online: 14.12.99.

FreeBSD (Hrsg.): The FreeBSD Inc., Deutschland, <URL: http://www.de.FreeBSD.org/de/>, online: 17.12.99.

FreeBSD (Hrsg.): The FreeBSD Project, <URL: http://www.freebsd.org/>, online: 17.12.99.

Free Software Foundation (Hrsg.): GNOME 1.0 - GNU Project, <URL: http://www.gnu.org/press/gnome-1.0.html>, online: 15.12.99.

Free Software Foundation (Hrsg.): GNU General Public License, <URL: http://www.fsf.org/copyleft/gpl.html>, online: 20.12.99.

Free Software Foundation (Hrsg.): GNU Lesser General Public License, <URL: http://www.gnu.org/copyleft/lesser.html>, online: 14.11.99.

Free Software Foundation (Hrsg.): GNU's Not Unix!, <URL: http://www.fsf.org/>, online: 22.10.99.

Free Software Foundation (Hrsg.): GNU's Who?, <URL: http://www.fsf.org/people/people.html>, online: 17.12.99.

Free Software Foundation (Hrsg.): Overview of the GNU Project, <URL: http://www.fsf.org/gnu-history.html>, online: 22.10.99.

Free Software Foundation (Hrsg.): Software, <URL: http://www.fsf.org/software.html>, online: 22.10.99.

Free Software Foundation (Hrsg.): What is free Software?, <URL: http://www.fsf.org/free-sw.html>, online: 22.10.99.

Furubotn, E. G.;Pejovich, S.: Property Rights and Economic Theory: A Survey of Recent Literature, in: Journal of Economic Literature, Vol. 10 (1972), S. 1137-1162.

Furubotn, E. G.;Pejovich, S. (Hrsg.): The Economics of Property Rights, (Ballinger) Cambridge, Mass. 1974.

Granovetter, M. S.: The strength of weak ties, in: AJS 78 (1973) S. 1360-1380.

Hackvän, S.: Reverse-engineering the GNU Public Virus, <URL: http://linuxworld.com/linuxworld/lw-1999-09/lw-09-gnu_p.html>, online: 26.11.99.

Haesler, A. J.: Tausch und gesellschaftliche Entwicklung, zur Prüfung eines liberalen Topos, Diss., St. Gallen 1983.

Hayek, F. A. von: Der Wettbewerb als Entdeckungsverfahren, Kiel 1968.

Hayek, F. A. von: The Use of Knowledge in Society, in: American Economic Review, 35 (1945), S. 519-530.

Hecker, F.:Setting Up Shop: The Business of Open-Source Software, <URL: http://people.netscape.com/hecker/setting-up-shop.html>, online: 11.12.99.

Heise Verlag (Hrsg.): US-Justizministerium: Microsoft in vier Punkten schuldig, <URL: http://www.heise.de/bin/nt.print/newsticker/data/hob-07.12.99-000/?id=de46e211&todo=print>, online: 07.12.99.

Heß, H.: Wiederverwendung von Software: Framework für betriebliche Informationssysteme, (Gabler) Wiesbaden 1993.

Hill, C. W.: Coorperation, Opportunism, and the Invisible Hand: Implications for Transaction Cost Theory, in: Acadamy of Management Review 15 (1990), S. 500-513.

Himstedt, T.: Python: Objektorientierte Scriptsprache fürs World Wide Web, <URL: http://www.heise.de/ix/artikel/9603144/#oberfl>, online: 26.02.96.

Homans, G. C.: Social Behaviour: Its Elementary Forms, New York 1974.

Homans, G. C.: Fundamental Social Processes, 1972.

Hubbard, J. K.: A Brief History of FreeBSD, <URL: http://www.freebsd.org/handbook/history.html>, online 17.12.99.

IBM (Hrsg.): Neuheiten für Linux, <URL: http://www-4.ibm.com/software/is/mp/linux/german.pdf>, online: 20.12.99.

IBM (Hrsg.): IBM helps companies turn simple web sites into powerful e-business solutions, <URL: http://www.ibm.com/News/1998/06/223.phtml>, online: 25.11.99.

Johanson, J.; Mattson, L. G.: Interorganizational Relations: A Network Approach Compared to the Transaction-Cost Approach, in: International Studies of Management and Organization 17 (1987) 1, S. 34-48.

Johnston, R.; Lawrence, P. R.: Beyond Vertical Intergation - The Rise of the Value-Adding Partnership, in: Harvard Business Review 66 (1988), S. 93-103.

KDE (Hrsg.): The K Desktop Environment, <URL: http://www.kde.org>, online: 15.12.99.

Kieser, A. (Hrsg.): Organisationstheorien, 3. Aufl., (Kohlhammer) Stuttgart et al. 1999.

Lash, A.: Evolution of a Net community, <URL: http://news.cnet.com/category/0-1005-201-326080.html>, online: 02.02.98.

Leo English/German Dictionary Team (Hrsg.): <URL: http://dict.leo.org/?search=community>, online:28.11.99.

Linux International (Hrsg.): Welcome to Linux International, <URL: http://www.li.org>, online: 15.12.99.

Linux Journal (Hrsg.): Linux Journal, <URL: http://www2.linuxjournal.com/cgi-bin/frames.pl/index.html>, online: 02.12.99.

Linux Online (Hrsg.): The Linux Homepage at Linux Online, <URL: http://www.linux.org/>, online: 19.12.99.

Losi, S.: Stephanie Chats with Uwe Thiem, <URL: http://go.borders.com/features/slo99092.xcv >, online: 21.12.99.

Loviscach, J.: Shareware und Freeware für Office, Internet, Dateimanagement und Systempflege, in: c't, 12(1999), S. 104-106.

Luhmann, N.: Soziologische Aufklärung 2. Aufsätze zur Theorie der Gesellschaft, (Westdeutscher Verlag) Opladen 1982.

Miles, R. E.; Snow, C. C.: Organizations: New Concepts for New Forms, in: California Management Review 28 (1986), S. 62-73.

Moody, G.: The Greatest OS That (N)ever Was, <URL: http://www.wired.com/wired/archive//5.08/linux.html?person=linus_torvalds&topic_set=wiredpeople>, online: 18.12.99.

Müller, M.: Die Philosophie des GNU und die Pragmatik des Open Source, in: O'Reilly & Associates, Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 17-19.

Müller, M.: Open Source-Projekte, in: O'Reilly & Associates, Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 21-31.

Netcraft (Hrsg.): Netcraft Web Server Survey, <URL: http://www.netcraft.com/survey/>, online: 15.12.99.

Netsape Inc. (Hrsg.): Press Release, <URL:

http://www.netscape.com/newsref/pr/newsrelease558.html>, online: 25.11.99.

OASE (Hrsg.): OASE: Was ist Shareware, <URL: http://members.aol.com/wwwoase/oase/defshw.htm>, online: 18.11.99.

OpenBSD (Hrsg.): OpenBSD: Multiplatform Ultra-secure OS, <URL: http://www.openbsd.org/>, online: 18.12.99.

OpenSource.Org (Hrsg.): Frequently Asked Questions about Open Source, <URL: http://www.opensource.org/faq.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): Halloween I: Open Source Software -- A (New?) Development Methodology, <URL: http://www.opensource.org/halloween/halloween1.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): Halloween II: Linux OS Competetive Analysis: The Next Java VM, <URL: http://www.opensource.org/halloween/halloween2.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): History of the Open Source Initiative, <URL: http://www.opensource,org/history.html>, online:12.11.99.

OpenSource.Org (Hrsg.): The Approved Licenses, <URL: http://www.opensource.org/licenses.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): Open Source: Software Gets Honest, <URL: http://www.opensource.org>, online: 25.11.99.

OpenSource.Org (Hrsg.): The Case for Open Source: Hackers' Version, <URL: http://www.opensource.org/for-hackers.html#marketing>, online:12.11.99.

OpenSource.Org (Hrsg.): The Open Source Definition, <URL: http://www.opensource.org/osd.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): The OSI Certification Mark and Program, <URL: http://www.opensource.org/certification-mark.html>, online: 12.11.99.

OpenSource.Org (Hrsg.): Why `Free Software' Is Too Ambiguous, <URL: http://www.opensource,org/free-notfree.html>, online:12.11.99.

O'Reilly & Associates Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999.

O'Reilly (Hrsg.): Welcome to the O'Reilly Perl Center, <URL: http://www.perl.oreilly.com/>, online 20.12.99.

Perl Mongers (Hrsg.): Perl Fast Facts, <URL: http://www.perlmongers.org/press/fast_facts.html>, online 20.12.99.

Picot A., Dietl H., Franck E.: Organisation: eine ökonomische Perspektive, (Schäffer-Poeschel) Stuttgart 1997.

Probst, G.: Selbstorganisation und Entwicklung, in: DU 41 (1987) 1987, S. 242.255.

Python (Hrsg.): Python Language Website, <URL: http:www.python.org>, online: 21.12.99.

Python (Hrsg.): Summary Information for HTTP Log, >URL: http://www.python.org/stats/>, online: 21.12.99.

Raymond, E.: A Brief History of Hackerdom, in: DiBona, C.; Ockmann, S.; Stone, M.

(Hrsg.): Open Sources: Voices from the Open Source Revolution, (O'Reilly) Sebastopol 1999, S. 19-29.

Raymond, E. (Hrsg.): Eric Steven Raymond's Home Page, <URL: http://www.tuxedo.org/~esr/>, online: 06.10.99.

Raymond, E.: How To Become A Hacker, <URL: http://www.tuxedo.org/~esr/faqs/hacker-howto.html>, online: 15.10.99.

Raymond, E. (Hrsg.): The Cathedral and the Bazaar, <URL: http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>, online: 23.09.99.

Raymond, E.: Open Source Evangelist, in: O'Reilly & Associates, Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 49-55.

SAMBA (Hrsg.): Samba Survey Manager, <URL: http://anu.samba.org/pub/samba/survey/ssstats.html>, online: 10.12.99.

Schumpeter, J. A.: Theorie der wirtschaftlichen Entwicklung, 6. Aufl., Berlin 1964.

Siebert, H.: Ökonomische Analyse von Unternehmensnetzwerken, in: Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 291-311.

Siebert, H.: Technologische Entwicklung und Vorproduktbeschaffung, Frankfurt 1990.

Schmidt, J.: Gemischtes Doppel: Linux und NT als Web-Server im Test, in: c't, 13(1999), S. 186-191.

Scholz, C.: Strategische Organisation: Prinzipien zur Vitalisierung und Virtualisierung, (Moderne Industrie) Landsberg/Lech 1997.

Scriptics (Hrsg.): Management Team, <URL: http://www.scriptics.com/company/people.html>, online: 21.12.99.

Scriptics (Hrsg.): Milestones, <URL: http://www.scriptics.com/company/news/>, online: 21.12.99.

Shailaja, V. R.: Linux in India, <URL: http://www.performancecomputing.com/columns/currents/9910cur.shtml>, online: 23.11.99.

Smith, A.: An Inquiry into the nature and causes of the wealth of nations, Oxford 1976.

Staehle, W. H. (Hrsg.): Managementforschung I, (de Gruyter) 1991.

Staehle, W. H.: Redundanz, Slack und lose Kopplung, in: Staehle, W.H. (Hrsg.): Managementforschung I, (de Gruyter) 1991, S. 313-345.

Stallman, R.: Richard Stallman on freedom and the GNU GPL, <URL: http://linuxworld.com/linuxworld/lw-1999-11/lw-11-rms_p.html>, online 26.11.99.

Stallman, R. (Hrsg.): Richard Stallman's Personal Home Page, <URL: http://www.gnu.org/people/rms.html>, online: 22.10.99.

Stallman, R.: The GNU Operating System and the Free Sofware Movement, in: DiBona, C.: Ockmann, S.; Stone, M. (Hrsg.): Open Sources: Voices from the Open Source Revolution, (O'Reilly) Sebastopol 1999.

SuSE (Hrsg.): LinuKS: SuSE Linux KDE Service,

<URL: http://www.suse.de/de/support/download/LinuKS/index.html>, online: 25.12.99.

The Apache Software Foundation (Hrsg.): Apache Group License, <URL: http://www.apache.org/license.txt>, online: 14.10.99.

The Apache Software Foundation (Hrsg.): The Apache Software Foundation, <URL: http://www.apache.org>, online: 18.12.99.

The Gimp (Hrsg.): The Gimp Homepage, <URL: http://www.gimp.org/the_gimp_about.html>, online: 22.12.99.

The Mozilla Organisation (Hrsg.): Mozilla.org, <URL: http:www.mozilla.org/>, online: 08.12.99.

The NetBSD Project (Hrsg.): About the NetBSD Project, <URL: http://www.netbsd.org/Misc/about.html>, online: 18.12.99.

The NetBSD Project (Hrsg.): Hardware Supported by NetBSD, <URL: http://www.netbsd.org/Ports/>, online: 18.12.99.

Thorelli, H. B.: Networks: Between Markets and Hierarchies, in: Strategic Management Journal, 7 (1986), S. 37-51.

Torvalds, L.: Der Pragmatiker der freien Software, in: O'Reilly & Associates, Inc. (Hrsg.): Open Source - kurz & gut, 1. Aufl., (O'Reilly) Köln 1999, S. 33-38.

Troll Tech (Hrsg.): Announcement: Open Source, <URL: http://www.troll.no/announce/qpl.html>, online: 12.12.99.

Troll Tech (Hrsg.): The QPL, an Open Source License, <URL: http://www.troll.no/qpl/>, online: 12.12.99.

Weick, K. E.: The Social Psychology of Organizing, 2. Aufl. (Addison Wesley) London 1979.

Williamson, O. E.: The Economic Institution of Capitalism: Firms, Markets, Relational Contracting, (Free Press) New York 1985.

Williamson, O. E.: Markets and Hierarchies: Analysis and Antitrust Implications, (Free Press) New York 1975.

ZDNet News (Hrsg.): Produktpreis „Innovation des Jahres" verliehen, <URL: http://www.zdnet.de/news/artikel/1999/03/19011-wf.htm>, online 19.03.99.

Zeep, W.: Redundanz - Ein Mittel zur Steigerung der Zuverlässigkeit von technischen Systemen, in: Bussmann, K. F.; Mertens, P. (Hrsg.): Operations Research und Datenverarbeitung bei der Instandhaltungsplanung, Stuttgart 1968, S. 83-102.