St. Ignucius von Emacs

Freie Software oder freier Markt?

Josef Templ

Software Templ OEG
Josef.Templ@aon.at,
http://members.aon.at/software-templ

Zusammenfassung Software wird zum überwiegenden Teil als Individualsoftware für einen bestimmten Kunden angefertigt. Die Entwicklung dieser Software ist eine Dienstleistung, die nicht beliebig vervielfältigt werden kann. Nur ein kleiner Teil (5 - 10%) ist Standardsoftware, die in Form von Nutzungsberechtigungen den Anwendern zur Verfügung gestellt wird. Nutzungsberechtigungen können beliebig vervielfältigt werden und stellen somit ein Geschäftsmodell dar, das auf künstlicher Verknappung eines Gutes basiert. In letzter Zeit ist dieses Modell ins Wanken geraten, und es stellt sich die Frage, woran das liegt und wie die Zukunft der Standardsoftware aussehen könnte. Dieser Artikel zeichnet überblicksmäßig die Entwicklung einer Alternative—Open Source Software—nach und möchte zum Nachdenken über zukünftige Geschäftsmodelle für Standardsoftware anregen.

1 Präambel

Meine Zeit am Institut von Prof. Rechenberg blieb mir in Erinnerung als beständige Suche nach neuen Trends und Denkmustern in der Softwareentwicklung. Wir haben uns neben den etablierten imperativen Programmiersprachen immer auch mit alternativen Konzepten wie funktionaler und logischer Programmierung beschäftigt. Mit Prolog haben wir experimentelle Compiler gebaut und Denksportaufgaben aus Zeitschriften gelöst. Objektorientierung war schon früh ein Forschungsthema, attribuierte Grammatiken ebenso. Versionsverwaltung und vieles mehr wurde aktiv untersucht und weiterentwickelt.

Dieser Beitrag, den ich meinem sehr geschätzten Lehrmeister Prof. Peter Rechenberg widme, berichtet überblicksmäßig von einer neuen Strömung, das im universitären Umfeld entstand und in den letzten 20 Jahren weitgehend unbeachtet von der Fachöffentlichkeit eine beachtliche Eigendynamik entwickelt hat. Es handelt sich dabei nicht um technische oder theoretische Errungenschaften, sondern um neue Perspektiven bezüglich der Organisation des Entwicklungsprozesses großer Softwareprojekte und um die Entwicklung zukünftiger Geschäftsmodelle für Standardsoftware.

Die genaue Bedeutung des Titels wird später im Artikel geklärt. Vorweg sei aber festgehalten, dass er keineswegs blasphemisch verstanden werden sollte und nichts mit einem real existierenden Heiligen zu tun hat. Der Titel trägt

vielmehr dem Umstand Rechnung, dass viele Erkenntnisse der Softwareentwicklung (leider) mit fast religiösem Eifer vertreten werden, obwohl eine sachliche Betrachtung, die im Untertitel zum Ausdruck kommt, zweckdienlicher wäre.

Dieser Artikel verwendet Material aus Dokumenten, die unter der GNU Free Documentation License (GNU FDL) [GFDL] stehen, wodurch dieser Artikel selbst unter die GNU FDL fällt.

2 Papierstau

Anfang der 80-er Jahre steht der 27-jährige Richard M. Stallman, Harvard Absolvent, talentierter und begeisterter Programmierer (damals noch im positiven Sinn *Hacker* genannt) im renommierten MIT AI Lab vor einem brandneuen Laserdrucker und wartet auf den Ausdruck eines 50-seitigen Dokuments. Sein Warten ist vergeblich. Der Drucker leidet unter *Papierstau*. Nach dessen Behebung kommen vier Seiten heraus, allerdings von einem viel früher übermittelten Dokument eines anderen Benutzers. Stallman wartet weiter und macht sich Gedanken über die Behebung des Problems. Da der Papierstau ein mechanisches Problem ist, bietet sich als Lösung nur der schon früher verwendeter Ansatz an, nämlich die Benachrichtigung der wartenden Benutzer, damit ein Stau so rasch wie möglich behoben werden kann. Stallman ist voller Zuversicht, dass damit auch in diesem Fall eine praktikable Lösung des Problems gefunden wird, obwohl zur Benachrichtigung die im Drucker eingebaute Software erweitert werden muss und die Quelltexte dafür nicht verfügbar sind. Man kennt sich in der Branche und der Zugriff auf die Quelltexte war noch nie ein Problem. Es gelingt ihm auch, entsprechende Kontakte zu anderen Forschern zu knüpfen, die Zugang zum Quelltext besitzen.

Die Ernüchterung könnte größer nicht sein, als Stallman feststellt, dass seine Kontaktpersonen ihm zwar helfen wollen, aber nicht helfen dürfen. Stallman wird das Opfer einer neuen Regelung in der Softwareentwicklung, nämlich des *Non Disclosure Agreements* (NDA). Er könnte als (privilegierter) MIT AI Forscher nun auch ein NDA unterzeichnen, erkennt aber, dass dadurch andere Kollegen, die von ihm Hilfe in dieser Sache benötigen, zu *seinen* NDA-Opfern würden. Stallman sieht eine unheilvolle Entwicklung auf die Softwareentwickler zukommen und beginnt sich aus tiefer Überzeugung dagegen zu stemmen.

3 Softwarelizenzen

In der darauffolgenden Zeit wurde Standardsoftware mehr und mehr zum kommerziellen Produkt und der Verkauf von Nutzungsberechtigungen (Lizenzen)wurde das dominierende Geschäftsmodell. Der Quelltext wurde streng unter Verschluss gehalten und man sprach von *proprietärer* Software. Viele Kollegen von Stallman wechselten die Seiten und nahmen gut bezahlte Jobs bei Firmen an, die proprietäre Software herstellten. Viele dieser Firmen gingen an die Börse, um das schnelle Wachstum zu finanzieren und die neue Geschäftsidee in klingende Münze umzusetzen. Standardsoftware, die beliebig leicht vervielfältigt werden

kann, versprach hohe Renditen für die Anleger. Man musste nur die Entwicklung finanzieren und mit großem Marketingaufwand eine noch größere Anzahl von Lizenzen verkaufen. Für einige Firmen war dieser Ansatz sehr erfolgreich, für andere zeigte sich aber bald ein unangenehmer Effekt.

4 The Winner Takes It All

Der hohe Einsatz für die Entwicklung immer komplexerer Anwendungen und die Marketingausgaben für einen globalen Vertrieb führten zu einer Monopolisierung des Marktes, weil sich der Einsatz nur noch für den Marktführer lohnte. Die Monopolisten sicherten ihre jeweiligen Reiche durch künstlichen Barrieren ab, die den Markteintritt kleinerer, technisch oft überlegener Konkurrenten erfolgreich abwehrten. Im folgenden seien nur einige Beispiel dafür genannt.

Das Patentrecht, das in manchen Ländern auch auf Software anwendbar ist, wurde als strategische Waffe eingesetzt. Die Komplexität der Anwendungen wurde nach Kräften erhöht. Inkompatibilitäten zu bestehenden Standards wurden gezielt eingeführt. Betriebssystemfunktionen wurden nicht oder nur unvollständig dokumentiert. Knebelungsverträge zur Ausweitung bestehender Monopole auf neue Gebiete wurden den Händlern aufgezwungen.

Dies und wahrscheinlich vieles mehr diente zur Erreichung und Verteidigung eines de-facto-Monopols in einem bestimmten Anwendungsbereich und führte zum weitgehenden Zusammenbruch dieses Geschäftsmodells und zum Platzen der Träume vieler Investoren. Der dramatische Verfall der Aktienkurse vieler Softwarefirmen und das Ausbleiben von Risikokapital für Neugründungen ist die sichtbare Folge dieser Entwicklung.

5 GNU

Unbeeindruckt von dieser Entwicklung arbeitete Stallman weiter an seiner Vision von freier—im Gegensatz zu propriäterer—Software und stellte den durch Lisp-Funktionen erweiterbaren Texteditor EMACS der Allgemeinheit zur Verfügung. Die große Anwendergemeinde trug beständig zur Weiterentwicklung, Verbesserung und Portierung auf neue Plattformen bei. Daneben begann die Vision eines vollständig frei verfügbaren Betriebssystems einschließlich wichtiger Werkzeuge Gestalt anzunehmen. Dieses System sollte auf Unix basieren aber zu 100% frei sein und garantiert für alle Zeit frei bleiben. Dazu arbeitete Stallman zunächst an einem C-Compiler, der für verschiedene Rechnerarchitekturen angepasst werden konnte, und an einem Debugger. Der Name des Mammutprojektes: *GNU*, ein rekursives Akronym für *G*nu's *N*ot *U*nix. Durch die viele Tipparbeit zog sich Stallman Gelenks- und Sehnenleiden zu, und musste zeitweise die Programme einer Schreibkraft diktieren. Heute verwendet er eine sehr leichtgängige Tastatur mit besonders kleinem Tastenhub, um die Hände zu schonen.

Finanziert wurde die Arbeit unter anderem durch Spenden, die von den Anwendern kamen. Zur Organisation und Verwaltung des Spendenflusses für das GNU-Projekt wurde der gemeinnützige Verein *Free Software Foundation* [FSF]

gegründet. Um die Ergebnisse des GNU-Projekts für die Öffentlichkeit zu bewahren und eine spätere Umwandlung in proprietäre Software zu unterbinden, schuf Stallman die *GNU General Public License* (GNU GPL) [GPL].

6 GPL

Frei verfügbare Software gab es schon vor Stallman und auch neben ihm[1], aber kaum jemand machte sich Gedanken darüber, wie man diese Freiheit im Detail gestalten und welche Arten von Nutzung man erlauben sollte. Mit der GPL schuf Stallman einen Vertrag, der die freie Nutzung seiner Software erlaubte. Ebenso erlaubte er die Modifikation und Weitergabe der Quelltexte, aber mit der Bedingung, dass jedes abgeleitete Werk selbst wieder der GPL unterliegt. Diese Eigenschaft ist der Kern der GPL und wird manchmal auch als deren *viraler Charakter* bezeichnet. Die GPL erlaubt explizit die Einhebung von Gebühren für Dienstleistungen im Zusammenhang mit freier Software, fordert aber, dass das Ausführen eines Programms ohne Einschränkungen möglich ist. Das Benutzungsrecht wird nicht als knappes und damit wirtschaftliches Gut gesehen.

Da im Sinne der GPL auch das Binden eines Programms mit einer unter GPL stehenden Bibliothek ein abgeleitetes Werk darstellt, gibt es noch eine abgeschwächte Form davon, die *GNU Lesser General Public License* (GNU LGPL). Das Binden eines Programms mit einer LGPL-Bibliothek stellt kein abgeleitetes Werk dar und ermöglicht somit die Erstellung von proprietärer Software auf der Grundlage von freien Bibliotheken.

7 Linux

Ende der 80-er Jahre fehlte vom GNU-Projekt nur noch ein Teil, der *Kernel*. Es war geplant, den Betriebssystemkern auf einer modernen Microkernel-Architektur (Mach) aufzubauen. Das dazu ins Leben gerufene Projekt (HURD) kam aber nur sehr langsam voran, weil die GNU-Programmierwerkzeuge für das Entwicklen und Debuggen einer Microkernel-Architektur nicht ausgelegt waren und umfangreiche Anpassungen erforderten.

Weit entfernt von Stallman entwickelt der Informatikstudent Linus Torvalds in Finnland als Hobby-Projekt einen Unix-Kernel mit einer traditionellen monolithischen Architektur. Nach einiger Entwicklungszeit stellte er einen ersten Prototyp im Internet zur Verfügung—Linux erblickte das Licht der Welt. Eine interessierte Anwendergemeinde formierte sich, forcierte die Weiterentwicklung und führte dazu, dass Linus Torvalds sein Linux unter die GNU GPL stellte. Linux wurde dadurch zu GNU/Linux, dem fehlenden Teil von Stallmans propagiertem freien Betriebssystem.

[1] Zum Beispiel wurde am Institut von Prof. Rechenberg in den 80-er Jahren der Parsergenerator Coco [Coco] entwickelt und nach eingehender Diskussion inklusive Quelltext frei verfügbar gemacht.

Linux stellt eigentlich keine technische Neuerung dar und Andrew Tanenbaum kritisierte sogar die unzeitgemäße monolithische Architektur scharf. Der Beitrag von Linux liegt vielmehr in der Art, *wie* es entstand, und dem Umstand, dass es überhaupt existiert. An der Entwicklung vom primitiven Prototyp bis zum heutigen stabilen, effizienten und vielseitigen Betriebssystem haben unzählige Personen mitgewirkt, die koordiniert werden mussten. Eric Raymond vergleicht in [Ray99] die Linux-Entwicklung mit einem Basar und den traditionellen Entwicklungsstil mit einer Kathedrale. Der Basar ist gekennzeichnet durch ein Gewühl von Personen und Meinungen, die sich mehr oder weniger selbst organisieren, während der Bau einer Kathedrale von einem Baumeister zentral gesteuert wird. Linux verwendete als erstes im großen Stil das neue Medium Internet als Plattform für die Entwicklung eines großen Software-Projekts. Die erfolgreiche Anwendung des Basar-Stiles steht in gewisser Weise im Widerspruch zu etablierten Prinzipien der Softwareentwicklung, wie sie etwa von Brooks in [Bro75] formuliert wurden, und wird vielfach als wichtiger Beitrag zum Software-Engineering der 90-er Jahre angesehen.

8 Open Source

Mit der Tendenz zu Monopolen bei proprietärer Software gewann auch der Gegentrend der freien Software an Boden und wurde in breiteren Kreisen bekannt. Dabei zeigte es sich, dass der Begriff *freie Software* missverständlich ist. Er kann im Sinn von *gratis* aufgefasst werden, oder im Sinn von *Freiheit*. Stallmann fasste ihn primär im Sinn von Freiheit auf, was auch im Titel der Stallman-Biographie [Wil02] zum Ausdruck kommt. Viele IT-Entscheidungsträger fassten ihn aber als *Freibier für alle* auf und sahen die Marktwirtschaft als solche in Gefahr. Um dieser Mehrdeutigkeit entgegenzuwirken, wurde im Februar 1998 vor allem durch das Bestreben der Linux-Gemeinde der Begriff *Open Source Software* (OSS) geboren und die *Open Source Initiative* (OSI) [OSI] gegründet. Damit war die Grundlage für eine breitere Akzeptanz von freier Software in höheren IT-Managementkreisen gelegt. Die schnelle Verbreitung von Linux und anderer Open Source Software bestätigte die Richtigkeit dieser Entscheidung.

Stallman blieb auch von dieser Entwicklung unbeeindruckt und verwendet weiterhin den Begriff *freie Software*. Das einzige, was sich für ihn geändert hat, ist die Art der Tätigkeit. Er wird nun immer öfter als Redner zu Konferenzen und ähnlichen Veranstaltungen eingeladen und hat dadurch Gelegenheit, seinen ganz speziellen Sinn für Humor, wie in Abbildung 1 dargestellt, zu präsentieren.

9 Open for Business

Neben GNU/Linux wurde in praktisch allen Bereichen eine wahre Flut von Open Source Projekten gestartet, und es ist mittlerweile unmöglich, den Überblick zu bewahren. Es entstanden spezialisierte OSS-Portale im Internet, deren größtes (Source Forge) bereits mehr als 40.000 Projekte beherbergt. Die meisten OSS-Projekte verwenden CVS, ein OSS-Versionsverwaltungssystem, zur Verwaltung

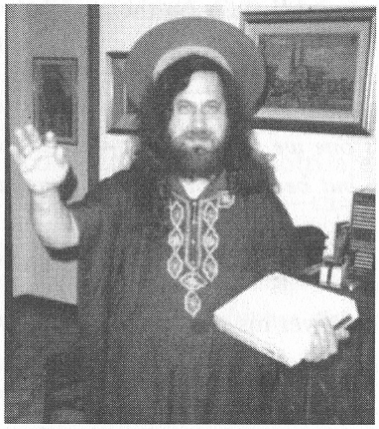

Abbildung 1. R. Stallman mit reflektierender Speicherplatte am Kopf und einer gelben Harddisk in der Hand: '*I am St. Ignucius of the Church of Emacs, I bless your computer my child.*'

der Quelltexte. Source Forge basiert auf Linux und Apache Web-Server, zwei OSS-Komponenten. Eine kleine Auswahl von OSS-Projekten sei im Folgenden erwähnt. Nicht alle Projekte basieren auf der GNU GPL, und manche sind nur eine Annäherung an die Open Source Prinzipien.

Die Programmiersprache Java wird inklusive Quelltexten für Compiler und Bibliothek zur Verfügung gestellt. Die wichtigsten Java-Entwicklungswerkzeuge (NetBeans und Eclipse) sind OSS-Projekte. Neben Server-Maschinen und Entwicklungswerkzeugen wurde auch der Desktop-PC zum Ziel von OSS-Bemühungen. Mit OpenOffice steht ein OSS-Office-Paket zur Verfügung. Das Textverarbeitungssystem TEX (bzw. LATEX) war ein Pionier der OSS-Idee lange vor dem GNU-Projekt. Mit ComPiere steht eine betriebswirtschaftliche Gesamtlösung (ERP) zur Verfügung, das in manchen Belangen nicht einmal den Vergleich mit SAP scheuen muss. Apple hat große Teile von Mac OS X der Open Source Definition angenähert, und hofft, dadurch Marktanteile zurückzugewinnen.

10 Geschäftsmodelle

Hinter all diesen Projekten steckt die Frage, wie sie finanziert werden. Können Idealisten wie Stallman auf Dauer qualitativ hochwertige Software bereitstellen, ohne dass ihnen irgendwann einmal der (finanzielle) Atem ausgeht? Tatsächlich haben sich bereits einige Finanzierungsformen etabliert, und die Entwicklung nezer Geschäftsmodelle ist noch nicht abgeschlossen. Im Zusammenhang mit den Finanzierungsformen wurden maßgeschneiderte Open Source Lizenzen entwickelt, die sich mehr oder weniger weit von der GNU GPL entfernt haben.Im Folgenden sind einige Finanzierungsmodelle skizziert.

Idealismus Der Drang, sich selbst zu bestätigen und einen Beitrag zu leisten, ist für manche eine ausreichende Motivation, um Zeit in OSS zu investieren. Dieser Aspekt ist aber nicht planbar und die verfügbare Kapazität teilt sich auf sehr viele Projekte auf.

Selbstvermarktung OSS-Projekte werden verwendet, damit potentielle Kunden auf die Leistungsfähigkeit eines Entwicklers oder eines Entwicklerteams aufmerksam werden. Dieser Aspekt spielt vor allem in Entwicklungs- und Schwellenländern eine Rolle.

Forschung Universitäre Forschung, die durch öffentliche Mittel finanziert wird, ist traditionell eine der tragenden Säulen von OSS und eine der wenigen Quellen für technische Innovation. Zu einem gewissen Teil werden Forschungsprojekte auch von kommerziellen Forschungsinstituten als OSS bereitgestellt, meist aber nicht unter der GPL. OSS ist durch diesen Finanzierungskanal immer nahe am Stand der Technik.

Eigenbedarf Der Bedarf nach neuer oder modifizierter Software, die nicht zur Kerntätigkeit eines Unternehmens zählt, führt oft zur Freigabe der Ergebnisse als OSS, um den Aufwand für Wartung zu reduzieren oder ganz auszulagern und die Qualität durch die Rückmeldungen der vergrößerten Anwendergemeinde zu erhöhen.

Marketing OSS wird als Marketingmittel verwendet, um mit wenig Aufwand einen Markt aufzubauen. Im Sog von OSS werden herkömmliche Produkte und Dienstleistungen verkauft.

Sponsoring Speziell größere OSS-Projekte leben häufig von Sponsoren, die verschiedenste Zwecke damit verbinden. Als Sponsoren kommen in Frage:

- Konkurrenten, die das Geschäftsmodell des Monopolinhabers einer proprietären Software zu untergraben versuchen.
- Dritte, die sich durch Dienstleistungen für OSS-Nutzer ein Geschäft erhoffen.
- Großkunden, die sich durch vergleichsweise geringe Spenden für OSS hohe Lizenzgebühren für proprietäre Software ersparen wollen. Dazu gehören vor allem auch öffentliche Verwaltungen, die jeweils eine große Anzahl von Lizenzen pro Produkt benötigen.

Trademarking Dieses spekulative Modell basiert auf der Idee, dass eine populäre Open Source Software eine starke Marke etablieren kann. Die Software wird zwecks Aufbau der Marke frei verteilt, mit der Trademark wird gehandelt. Wer die Marke verwenden will, um unter diesem Namen Produkte oder Services zu vertreiben, muss eine Lizenzgebühr an den Markeninhaber bezahlen, der damit das OSS-Projekt finanziert. Beispiele für solche Sekundärprodukte wären Fanartikel, gedruckte Dokumentation oder Seminare.

11 Ausblicke

OSS beginnt als ernsthafte Bedrohung für proprietäre Software wahrgenommen zu werden, und es formieren sich Gegenstrategien. Dazu gehören unter anderem

Patente, proprietäre Hardware und das Zurückhalten von Forschungsergebnissen.

Das Patentrecht wird oft nicht als Schutz geistigen Eigentums, sondern gezielt als Stolperstein für Konkurrenten eingesetzt. Durch Patente auf triviale Sachverhalte, die auf wundersame Weise die Patentämter passieren, kann eine proprietäre Software sehr gut geschützt werden. Seit beteiligte Anwälte ihre Praktiken offengelegt haben, sind Softwarepatente zwar ins Zwielicht geraten, die EU-Kommission möchte Patente auf Software aber nach amerikanischem Muster einführen.

Fehlende Dokumentation von Hardwarebausteinen kann eine unüberwindliche Hürde für OSS sein. Viele Hersteller haben aber, bedingt durch die starke Verbreitung von Systemen wie Linux, ihre Schnittstellen offengelegt, um sich diesem Markt nicht zu verschließen.

Durch die immer stärker verbreitete Finanzierung oder Kofinanzierung von Forschungsarbeiten durch Drittmittel aus der Industrie ist die Publikation der Ergebnisse nicht mehr selbstverständlich und die Freigabe als OSS oft nicht mehr möglich.

An öffentliche Fördermittel für gewerbliche oder industrielle Forschungs- und Entwicklungsprojekte ist oft die Auflage gebunden, dass die Ergebnisse dem Förderungsnehmer einen unmittelbaren wirtschaftlichen Vorteil verschaffen müssen, was eine Förderung von OSS-Projekten praktisch ausschließt. Der mögliche volkswirtschaftliche Nutzen sowie mögliche Innovationen im Bereich der Entwicklung neuer Geschäftsmodelle werden dabei außer Acht gelassen.

Der Jurist des Cyberspace, Lawrence Lessig, sieht in [Les01] aber auch ein Problem mit proprietärer Software, und zwar mit der prinzipiellen Anwendbarkeit des Copyrights. Das Copyright ist seiner Ansicht nach ein Vertrag zwischen Autor und Öffentlichkeit, in dem die Öffentlichkeit dem Autor im Gegenzug zur Veröffentlichung eines Werkes ein Exklusivrecht darauf einräumt. Bei proprietärer Software sieht Lessig dieses Prinzip verletzt, weil keine Veröffentlichung des Werkes stattfindet. Eine Softwarefirma, die in Konkurs geht, könnte so den geschützten Inhalt des Werkes für immer der Öffentlichkeit vorenthalten. Die Öffentlichkeit könnte dadurch geschädigt werden, wenn zum Beispiel elektronische Dokumente später nicht mehr gelesen werden können, weil das Dateiformat nicht dokumentiert ist. Lessig schlägt vor, dass zumindest eine Hinterlegung des Quelltextes bei einer neutralen Stelle erfolgen müsste, damit das Copyright überhaupt anwendbar ist.

Ein anderer Aspekt ist die zunehmende Digitalisierung unseres Lebens. Der Umgang mit dem Computer wird zu einer grundlegenden Kulturtechnik, die auf gleicher Stufe wie der Umgang mit Papier und Bleistift steht oder zumindest in absehbarer Zeit stehen wird. Der Zugang zu dieser neuen Kulturtechnik sollte—und dafür gibt es einen breiten gesellschaftlichen Konsens—für alle offen sein, was aber nur bei drastisch gesenkten Kosten für elementare Standardsoftware wie Betriebssystem, Textverarbeitung oder Internet-Browser erreichbar ist. Am Beispiel des Internet ist dieser Trend klar erkennbar. Der Internet-Boom wurde nicht zuletzt durch einen frei verfügbaren Web-Browser (Mosaic) ausgelöst und

hat eine Kultur der freien Verfügbarkeit von Software zum Umgang mit digitalen Inhalten etabliert.

12 Zusammenfassung

Derzeit befinden wir uns—nach meiner persönlichen Einschätzung—in einer Situation in der das alte Modell für Standardsoftware nicht mehr funktioniert und die neuen Modelle noch nicht. In welche Richtung sich die Geschäftsmodelle weiterentwickeln, bleibt ein spannendes, interdisziplinäres Thema, das Techniker, Betriebswirte, Juristen und Politiker einbezieht. Als selbständiger Informatiker wünsche ich mir jedenfalls beides: die Verfügbarkeit von Software in Quellform und die Verfügbarkeit eines tragfähigen Geschäftsmodells dahinter, aus dem alle beteiligten Parteien einen Nutzen ziehen, also sowohl Softwareentwickler als auch Anwender. Die Antwort auf die Frage im Untertitel dieses Artiels sollte also lauten: Freie Software *und* freier Markt.

Literatur

[Bro75] Brooks, Fred P.: The Mythical Man-Month. Addison Wesley, 1975.

[Coco] H. Mössenböck: Compiler Generator Coco/R: `http://www.ssw.uni-linz.ac.at/Research/Projects/#Coco`.

[FSF] Free Software Foundation: `http://www.fsf.org/`.

[GFDL] GNU Free Documentation License: `http://www.gnu.org/licenses/fdl.html`.

[GPL] GNU General Public License: `http://www.gnu.org/licenses/gpl.html`.

[Les01] Lessig, Lawrence: The Future Of Ideas. Random House, 2001.

[OSI] Open Source Initiative: `http://www.opensource.org/`.

[Ray99] Raymond, Eric S.: The Cathedral & The Basar. O'Reilly, 1999.

[Wil02] Williams, Sam: Free As In Freedom. O'Reilly, 2002.