Technische Universität Berlin
Fachbereich Informatik
Institut für Angewandte Informatik
Prof. Dr. iur. Bernd Lutterbeck

SICHERHEIT IN DER INFORMATIONSTECHNOLOGIE UND PATENTSCHUTZ FÜR SOFTWAREPRODUKTE – EIN WIDERSPRUCH?

KURZGUTACHTEN

Erstellt im Auftrag des Bundesministeriums für Wirtschaft und Technologie

vorgelegt von der
Forschungsgruppe Internet Governance
Berlin, Dezember 2000

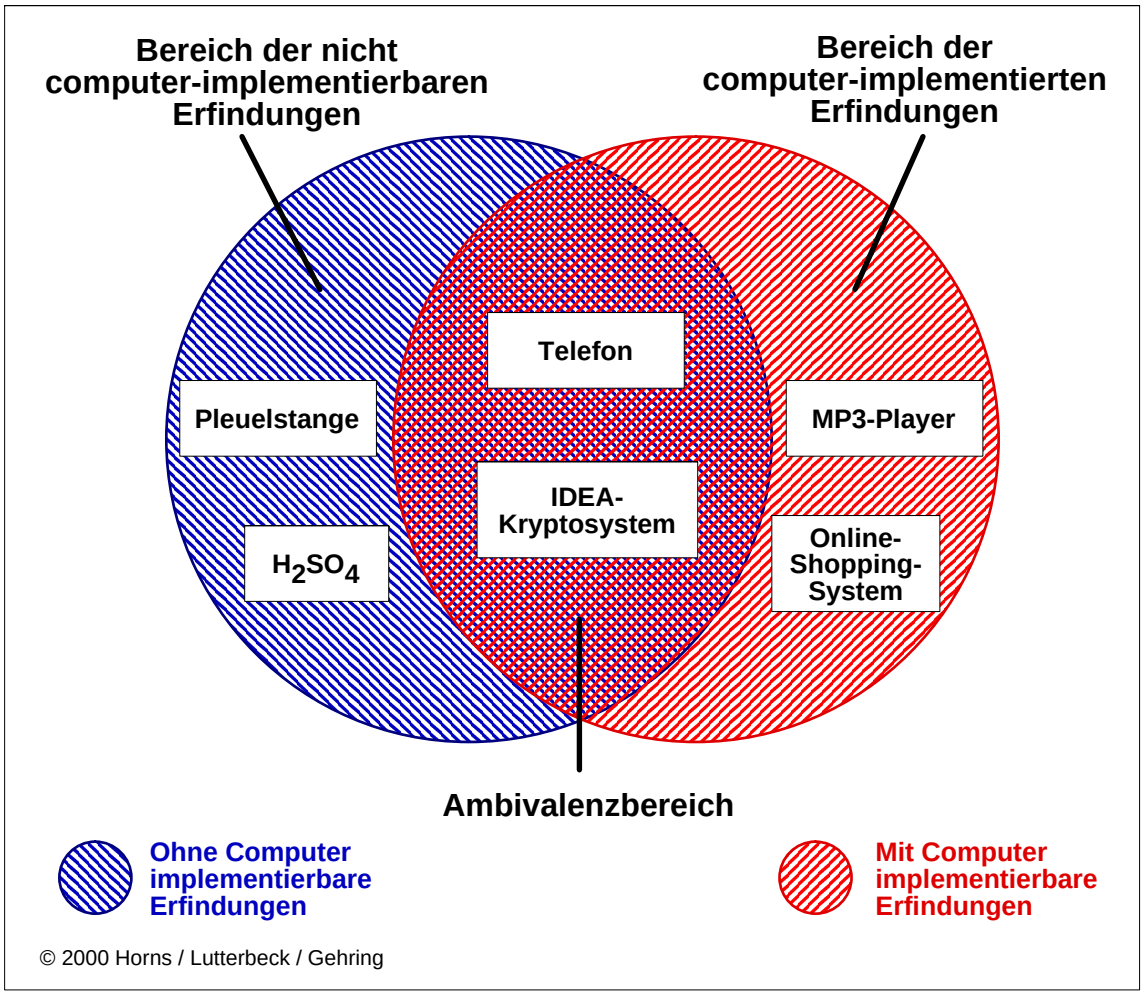
Bernd Lutterbeck (Professor für Informationsrecht, Jean Monnet Professor)
Robert Gehring (Diplom-Informatiker, Consultant, TU-Berlin)
Axel H. Horns (Diplom-Physiker, Patentanwalt in München)

Empfehlungen an die Politik

"Software−Patente" – ein Thema für die Geschichtsbücher?

Es gibt Fachleute, die die Zahl von erteilten "Software-Patenten" in Deutschland auf ca. 1200 für das Jahr 2000 schätzen[1], für die USA spricht der Patent Office Director von 1000.[2] Ob diese Zahlen zutreffen, lässt sich nicht mit Gewissheit feststellen: Es gibt weder eine allseits anerkannte Statistik "Software-Patente", noch besteht Konsens, wie man den Bereich der "Software-Patente" gegenüber anderen Erfindungspatenten abgrenzen muss. So lässt sich auch das überraschende Ungleichgewicht zwischen den deutschen und amerikanischen Zahlen gegenwärtig nicht aufklären.

Die folgende Zeichnung soll eine erste Vorstellung über die Struktur dieses Technikbereichs geben:



1 *Wolfgang Tauchert*, Leiter der Abteilung "Datenverarbeitung und Informationsspeicherung" im Deutschen Patent- und Markenamt, zit. nach **de Paole 2000**.

2 Einschließlich «software business-method patents», die der Direktor des amerikanischen Patentamtes mit 553 für 1999 angibt, zit. nach **Gross 2000**; weitere Zahlen bei **Cohen/Lemley 2000 (2001)**, p. 14 Fn 31.

Erläuterung der Grafik

In Anlehnung an die Sprachregelung des Sondierungspapiers der EU-Kommission[3] bezeichnen wir die durch den linken Kreis symbolisierte Menge als *ohne Computer implementierbare* Patentgegenstände. Der rechte Kreis symbolisiert diejenigen Patentgegenstände, die *mit Computer implementierbar* sind.

Die Schnittmenge bildet einen *Ambivalenzbereich* von solchen Patentgegenständen, diesowohl mit als auch ohne Computer implementierbar sind. Beispiele dafür sind etwa Erfindungen für Telefone oder Datenverschlüsselungsgeräte, die sowohl als reine Hardwareausführung, als auch mit Softwarekomponenten konstruiert werden können.

Der linke (einfach schraffierte) Kreisausschnitt grenzt die *ausschließlich ohne Computer implementierbaren Erfindungen* ab. Beispielsweise fällt eine durch eine Erfindung verbesserte Pleuelstange in diesen Bereich.

Der rechte (einfach schraffierte) Kreisausschnitt umfasst symbolisch alle Patentgegenstände, die zur Implementierung eines Computers bedürfen, die *computer-implementierten Erfindungen*. MP3-Player, die das patentgeschützte MP3-Verfahren zur Tondatenkomprimierung benutzen, fallen beispielsweise in diesen Bereich.

Bei einem Patent lassen sich anhand der erteilten Patentansprüche allenfalls der Bereich der ausschließlich ohne Computer implementierbaren Erfindungen einerseits und der Bereich der computer-implementierten Erfindungen sicher erkennen.

Bei Erfindungen im Ambivalenzbereich decken die mit dem Patent erteilten Patentansprüche sowohl reine Hardwarelösungen als auch Software-basierte Lösungen ab. Wegen dieser Unklarheiten lässt es sich nur schwer beantworten, ob ein Patent ein "Software-Patent" ist oder nicht.

Wir verwenden in diesem Text das Wort "Software-Patent", wenn wir auf Zusammenhänge verweisen, in denen das Wort benutzt wird.

Deutsche, europäische und amerikanische Gerichte haben den Patentschutz von Software längst anerkannt. Die Frage «Ist Software patentierbar?» ist also praktisch längst beantwortet - «a matter for the history books»[4].

Gleichwohl hat das Thema "Software-Patente" gegenwärtig Konjunktur, weil trotz aller Schwierigkei-

---

3 Die Patentierbarkeit computer-implementierter Erfindungen. Sondierungspapier der Generaldirektion Binnenmarkt der Europäischen Union vom 19.10.2000, im Internet: http://europa.eu.int/comm/internal_market/en/intprop/indprop/softde.pdf (24.10.2000); am gleichen Tag hat die Kommission ein einschlägiges Gutachten publiziert: **Hart/Holmes/Reid 2000**.

4 **Cohen/Lemley 2000 (2001)**, p. 1.

ten der Abgrenzung alle Beteiligten diesseits und jenseits des Atlantiks[5] von der großen praktischen und ökonomischen Relevanz des Themas ausgehen. Dabei bilden alte Argumente und einige möglicherweise neuartige Sachverhalte ein Gemisch, das es zu trennen gilt:

- Wohlbekannt sind Argumente, die den Zusammenhang von Patentschutz und Innovation ökonomisch begründen oder bezweifeln – im Allgemeinen wie in Bezug auf die Entwicklung von Software.[6]
  Möglicherweise neu sind die Antworten, die die wachsende Zahl sog. freier Softwarentwickler auf die alte Frage gibt.[7]

- Wohlbekannt ist eine Fachöffentlichkeit, die sich der praktischen und wissenschaftlichen Durchdringung des Themas annimmt.
  Möglicherweise neu ist die Art und Weise, in der mit Hilfe des Mediums Internet Meinungen gebildet werden – vorbei an der Fachöffentlichkeit und getragen von Akteuren, die fast nie Ökonomen oder Juristen, sondern häufig Technologen oder Informatiker sind.[8]

- Neu und in Deutschland noch nirgends diskutiert sind Zahlen über die weltweite Verteilung von "Open Source"-Entwicklern. Deutsche Entwickler bilden danach die zweitgrößte Gruppe. Insgesamt sind europäische Entwickler dominierend. Diese Befunde sind so bemerkenswert, dass eine Erklärung mit den bekannten ökonomischen Modellen zunächst versagt.[9]

- Ganz sicher neu sind Argumente, die Fachleute aus dem Bereich der Sicherheit der Informationstechnologie (IT-Sicherheit) für die Offenlegung von Software ins Feld führen.

Vor allem die zuletzt genannten Gesichtspunkte geben der aktuellen Debatte um computer-implementierbare Erfindungen ein Gesicht, das den Kern der neuen Ökonomie betrifft. Behauptet wird: Die neue (Internet-)Ökonomie steht und fällt mit der Lösung von Problemen der Sicherheit und Verlässlichkeit der Netzwerke und der dafür eingesetzten Software. Nur Software, deren Quelltext offengelegt ist und von Jedermann geprüft und weiterentwickelt werden kann, ermöglicht ein akzeptables Sicherheitsniveau, so behaupten diese Fachleute.

---

5 Im Oktober 2000 haben zwei Abgeordnete der Demokratischen Partei im Kongress der Vereinigten Staaten eine Gesetzesinitiative eingebracht, die sich vor allem gegen die ausufernde Patentierung von Internetgeschäftsmodellen richtet: 106th Congress 2nd Session, H.R. 5364 "Business Method Patent Improvement Act of 2000" vom 3.10.2000, im Internet: http://www.house.gov/berman/HR5364.pdf (31.10.2000); vgl. auch das Statement des Abgeordneten *Berman* anlässlich der Einführung des Gesetzes, **Berman 2000**.

6 Gegenwärtig wird eine ökonomische Studie des MIT intensiv diskutiert, die den behaupteten Zusammenhang verneint, vgl. **Bessen/Maskin 2000**.

7 Es gibt noch keine wirklich harten Zahlen. Einigermaßen verlässlich dürften die Zahlen der empirischen Untersuchung von **Demsey/Weiss/Jones/Greenberg 1999** sein. Sie schätzen die Zahl der "Open Source"-Entwickler auf 250.000 Personen weltweit.

8 Deutsche Juristen übersehen häufig, dass die wichtigste Lizenz für quellenoffene Software von dem Informatiker *Richard Stallmann* und nicht von Juristen entwickelt wurde.

9 Über die Studie von **Demsey/Weiss/Jones/Greenberg 1999** berichtete erstmals **Glascock 2000** am 1.11.2000. Unter der Überschrift «Germany Leads In Open-Source Development» zitiert er einen Autor der Studie mit den Worten: «We knew the Germans were really active, but we didn't know how active. We were really knocked out when we saw the Germans were the second largest contributors.»

Sollte diese Sicht zutreffen, scheint es zwingend, dass selbst wohlbekannte Argumente nochmals auf den Prüfstand müssen.[10]

Wir werden in diesem Kurzgutachten die verschiedenen Argumente nicht in der gebotenen Tiefe behandeln können. Unser Ziel ist es lediglich, plausible Argumente für die folgenden Empfehlungen an die Politik zu entwickeln. Wir hoffen, dass es so möglich wird, ein altes − fast möchte man sagen altmodisches − Thema nochmals mit anderen Augen zu sehen. Beginnen sollten wir mit einer genaueren Sprechweise über das Thema "Software-Patente". Zwei führende amerikanische Akademiker auf dem Gebiet des geistigen Eigentums haben das in einem großen Aufsatz treffend so ausgedrückt:

> *«It is wrong to speak of a commercial program as being "patented" in the same sense that we might say it is "copyrighted". More properly, the software vendor has patents that cover certain inventions contained in the program. Many parts of the program, however, are unpatented.»*[11]

Bei der politischen Entscheidung über kleinere und größere Streitfragen mag es hilfreich sein, die Worte eines der großen Kenners des Patentrechts zu erinnern. Am Ende einer ökonomischen Betrachtung des Patentsystems resümiert *Fritz Machlup* im Jahr 1958:

> *«[...] if we did not have a patent system, it would be irresponsible, on the basis of our present knowledge of its economic consequences, to recommend instituting one. But since we have had a patent system for a long time, it would be irresponsible, on the basis of our present knowledge, to recommend abolishing it.»*[12]

Wir schlagen vor, dass die in diesem Statement zum Ausdruck kommende Haltung Richtschnur auch für die Politik heute sein sollte: Einerseits ist es zwingend, sich Vorstellungen zu machen, die über die Bipolarität des Gewerblichen Rechtsschutzes hinausgehen.[13] Dieses System ist ein Kind des 19. Jahrhunderts. Andererseits ist es zwingend, dieses System nicht überstürzt aufzugeben - mit völlig ungewissen Ausgang. Aus dieser Haltung heraus ist auch der «Berliner Ansatz zu "Open Software Patents"» entstanden.[14]

---

10 Man sollte jedoch die Warnung von *Bruce Schneier* vor logischen Kurzschlüssen nicht überhören: «I´m a fan of open source, and believe it has the potential to improve security. But software isn´t automatically secure because it is open source, just as it isn´t automatically insecure because it it is proprietary.» (**Schneier 2000**, p 345). *Schneier* ist ein weltweit führender und geachteter Experte der IT-Sicherheit.

11 **Cohen/Lemley 2000** (2001), p 41.

12 *Fritz Machlup* zit. nach **Kitch 1998** in seiner Kommentierung des Stichworts "patents" in Palgrave: Dictionary of Economics and the Law, 1998.

13 Seit dem Symposium an der Columbia University von 1994 ist diese Einsicht in der akademischen Welt der USA im Prinzip nicht mehr strittig, vgl. **Manifesto 1994** und **Reichman 1994**; in der Sache ähnlich, aber zurückhaltender der Bericht des National Research Council von 2000, vgl. **Digital Dilemma 2000**. Natürlich stellen sich auch ganz schwierige verfassungsrechtliche Fragen. Das Bundesverfassungsgericht hat dieses Problem im Zusammenhang mit dem Urheberrecht vor Kurzem erstmals ausführlicher behandelt, vgl. **Bundesverfassungsgericht 1 BvR 825/98 v. 29.6.2000** [Brecht Erben gegen Kiepenheuer & Witsch], im Internet über: http://www.bverfg.de (31.10.2000).

14 **Gehring 2000**; *Axel H. Horns* hat in einer Reihe neuer Aufsätze Konturen einer modernen Patentpolitik vorgestellt, vgl. **Horns 2000**, **Horns 2001**.

Empfehlungen

Im Einklang mit internationalen Regelwerken und technischen Normen unterscheiden wir drei Typen von Empfehlungen.[15]

Symbole

Bei den Empfehlungen werden Sie links verschiedene Symbole finden. Diese Symbole deuten auf die Klassifizierung der Empfehlungen hin.

★★★ **Risiko (Muss)**

Diese Empfehlung weist auf entscheidende Schlüsselsituationen für Formulierung und Umsetzung der Patentpolitik hin. Bei der Umsetzung dieser Empfehlung muss überlegt vorgegangen werden, da es sonst zu Problemen oder Fehlentscheidungen kommen könnte. Andernfalls können unbedachte Aktionen leicht große Folgen haben.

★★ **Wert / Wichtigkeit (Sollte)**

Die zugehörige Empfehlung zu diesem Symbol kennzeichnet wichtige Informationen, die für die Umsetzung der Patentpolitik von maßgeblicher Bedeutung sind. Die hier beschriebenen Empfehlungen sollten in jedem Fall beachtet werden.

★ **Hinweis (Wünschenswert)**

Diese Empfehlung ist nachgeordneter Natur, da eine höhere Klassifizierung erst möglich ist, wenn entsprechende Veränderungen der Patentpolitik (oder auch Veränderungen des Urheberrechts) bereits realisiert sind und ausreichend lange Erfahrungen mit neuen Regularien vorliegen. Trotzdem ist ihre Erwähnung notwendig, da sie bereits zum gegenwärtigen Zeitpunkt in die Lösungsansätze mit eingebunden werden müssen, um bei Abschätzung auf der Basis des jetzigen Wissens im Hinblick auf die Potenziale der angestrebten technischen Konzepte die Risiken und Aussichten umfassend zu erhalten.

---

15 Wir übernehmen die Typik des soeben fertig gestellten Projektberichts der Europäischen Akademie Bad Neuenahr−Ahrweiler über "Elektronische Signaturen. Kulturelle und moralische Beherrschbarkeit." [**Langenbach 2001**] *Bernd Lutterbeck* ist einer der Verfasser dieses Berichts.

Generelle Empfehlungen

★★★ **GE−1:** Die künftige Patentpolitik muss eine angemessene Balance zwischen Patentrecht, Urheberrecht und vor allem dem Verfassungsrecht herstellen. Wo dieser Zusammenhang übersehen wird, besteht die Gefahr, dass der Kern der künftigen Informationsgesellschaft ökonomisch fehlgesteuert wird.

Begründung:

Das Wissen in all seinen Erscheinungsformen stellt die wichtigste Ressource für die Ökonomie der Informationsgesellschaft dar. Das klassische Patent- und Urheberrecht entziehen diese Ressource in gewissem Umfang dem Einfluss der Marktkräfte, was unter der Bedingung der Ressourcenknappheit, d.h. der Wissensknappheit, sozial und wirtschaftlich vertretbar erscheint. Der Aufbau einer ganzen Wissensökonomie ohne die selbstregulierenden Kräfte des Marktes aber ist nach dem gegenwärtigen Wissensstand weder wünschenswert noch möglich.

★★★ **GE−2:** Das überkommene Patentsystem begünstigt Großunternehmen. Die Politik muss das wettbewerbsrechtliche Umfeld so gestalten, dass kleine und mittlere Unternehmen eine faire Chance haben, am Wettbewerb teilzuhaben.

Begründung:

Kleine und mittlere Unternehmen können vielleicht noch bei der Erteilung von Patenten im Wettbewerb bestehen. Die Marktverfolgungskosten und z.B. die Kosten für die Aufrechterhaltung von Patenten übersteigen jedoch die Möglichkeiten der meisten betroffenen Unternehmen bei Weitem. Dies ist insbesondere für die Bundesrepublik, aber auch für andere Mitglieder der Europäischen Union relevant, wo Software überwiegend von kleinen und mittleren Unternehmen entwickelt wird.

Es muss im Wettbewerbsrecht strikt darauf geachtet werden, dass ein allgemeiner Grundsatz nicht in Vergessenheit gerät: Wer Patente für sich in Anspruch nimmt, muss belegen, dass ein solches Monopol aus überwiegenden Gründen des Gemeinwohls gerechtfertigt ist. Es ist nicht Sache von "Open Source"-Unternehmen zu belegen, dass Patente ihr unternehmerisches Vorgehen behindern.

★★ **GE−4:** Das in Deutschland und Europa vorhandene kreative Potential an "Open Source"-Entwicklern stellt einen schützenswerten Standortvorteil im Bereich der IT-Industrie dar. Softwareentwicklung im "Open Source"-Prozessß trägt der Struktur der IT-Industrie in Europa − viele kleine und mittlere Unternehmen ohne Markführerschaft, erheblicher Mangel an ausgebildeten Fachleuten − optimal Rechnung und sollte umfassend gefördert werden.

Begründung:

Die jetzt bekannten Zahlen über die weltweite Verteilung freier Softwareentwickler (siehe Fussnote 9) lassen vermuten, dass in Deutschland ein besonders hohes Potential kreativer Entwickler vorhanden ist. Diese Personen gehen neue Wege und vergrößern dadurch die Wettbewerbsfähigkeit der deutschen Wirtschaft entscheidend. Die Zahlen belegen im Übrigen, dass es falsch ist, diesen Personenkreis als "Spinner", "naive Träumer" oder gar "Anarchisten" zu bezeichnen, die die Eigentumsordnung abschaffen wollen. Das Gegenteil ist der Fall – Ausnahmen bestätigen wie immer die Regel.

Das im "Open Source"-Bereich vorhandene Potential sollte gefördert werden. Geeignete Fördermaßnahmen des öffentlichen Bereichs könnten sein:

- Zielgerichtete Integration von "Open Source"-Prozessen in die IT-Ausbildung an den Hochschulen und in der Berufsausbildung.
- Schaffung der Voraussetzungen zur angemessenen Berücksichtigung von "Open Sorce"--Alternativen bei der Softwarebeschaffung im öffentlichen Bereich, z.B. durch Anpassung der Ausschreibungsverfahren und anbieterneutrale, öffentliche Auftragsversteigerungen.
- Finanzielle Förderung von Pilotprojekten, Anschub- oder Zwischenfinanzierung laufender Projekte von öffentlichem Interesse.

Geeignete Fördermaßnahmen der Privatwirtschaft könnten sein:

- Kooperatives, liberales Verhalten beim Informationsaustausch mit "Open Source"-Entwicklern, z.B. durch die Internet-Veröffentlichung von Produktspezifikationen.
- Einrichtung von Kontaktstellen für "Open Source"-Entwickler in den Unternehmen.
- Entwicklung von geeignetem Aufklärungsmaterial für die Industrie, Initiierung eines kontinuierlichen Erfahrungsaustauschs über den Einsatz und die Entwicklung von "Open Source"-Software.
- Stärkere Berücksichtigung offener Standards bei der Entwicklung neuer Produkte.
- Teilfinanzierung von "Open Source"-Projekten durch mehrere Unternehmen.

Allgemeine Empfehlung zur IT-Sicherheitspolitik

★★★ **IT−1:** Ein zukünftiger (Patent)Schutz für «computer-implementierte bzw. computer-implementierbare Erfindungen» muss die Belange der IT-Sicherheit angemessen berücksichtigen. Das volkswirtschaftliche Interesse an prüfbar sicherer, d.h. quellenoffener Software darf nicht durch bloßen Verweis auf die rechtliche Systematik vernachlässigt werden.

Begründung:

Je mehr Software an den Schaltstellen der Wirtschaft zum Einsatz kommt, je mehr Sorgfalt ist bei der Auswahl und dem Einsatz der Software geboten. Das öffentliche Interesse an sicherer Software muss betriebswirtschaftlichen Interessen einzelner Softwarehersteller an universeller Durchsetzung von Patenten vorgehen. Der Belohnungsmechanismus des Patentrechts hat bisher keinen Anreiz zur Entwicklung sicherer Software geboten. Sicherheit muss aber in Zukunft eines der entscheidenden Kriterien im gesellschaftlichen Umgang mit Software werden.

- Der Prozess der quellenoffenen Softwareentwicklung eröffnet konkrete Möglichkeiten für Wirtschaft, Staat und Wissenschaft, die Erstellung von Software unter Sicherheitsaspekten zu fördern. Die Erkenntnisse der IT-Sicherheitsexperten können bei der Systemgestaltung von Anfang an berücksichtigt werden.

- Erprobte Konzepte wie die aus dem Handelsrecht bekannten Grundsätze ordnungsgemäßer Datenverarbeitung liefern geeignete Ansatzpunkte für den sicherheitsbewussten Umgang mit Software und Daten.

- Die Quellenoffenheit der Software erscheint als ein geeignetes Instrument, das Vertrauen der Öffentlichkeit in diese Basistechnologie auf dem Weg in die Informationsgesellschaft zu fördern.

- Die Hersteller von Software werden durch die Überprüfbarkeit angehalten, Sicherheitskonzepte bei der Entwicklung ihrer Produkte stärker als bisher zu berücksichtigen.

- Vergleichbarkeit bei quellenoffener Software macht aus Sicherheit ein nachfragereguliertes Qualitätsmerkmal im Softwaremarkt. Die Entwicklung sicherer Software kann somit über den Marktmechanismus belohnt werden.

- Prüfbar sichere Software ist eine unabdingbare Voraussetzung, um den Erfordernissen des modernen Datenschutzes gerecht zu werden.

Empfehlungen zur Patentpolitik

★★★ **PP−1:** Der Umgang mit dem Quelltext von Computerprogrammen muss patentrechtlich privilegiert werden. Das Herstellen, Anbieten, in Verkehr bringen, Besitzen oder Einführen des Quelltextes eines Computerprogrammes in seiner jeweiligen Ausdrucksform muss vom Patentschutz ausgenommen werden. (**Quelltextprivileg**)

Begründung:

Diese Empfehlung muss als oberster, schlechthin unverzichtbarer Grundsatz alle Bemühungen um die künftige Patentpolitik leiten. Über den Sicherheitsgewinn hinaus setzt dieser Vorschlag die richtigen ökonomischen Anreize:

- Die Entwickler von "Open Source"-Software vermeiden das Risiko einer Patentverletzung.
- Der nichtgewerbliche Anwender beispielsweise beim privaten Gebrauch darf die von den privilegierten Entwicklern und Distributoren bereitgestellten Quelltexte ohne patentrechtliche Einschränkungen nutzen.

  Der gewerbliche Anwender fällt unter das Patent und muss für den Gebrauch des Programmes die Zustimmung des Patentinhabers einholen.
- Softwareunternehmen bekommen einen Anreiz, den Quelltext offenzulegen, um zumindest für Entwicklung und Distribution in den Genuss der Privilegierung zu kommen.

★★ **PP−2:** Es soll im Patentrecht keine Unterscheidung zwischen konventionellen Patenten und "Software-Patenten" vorgenommen werden. Insbesondere sollen weder die Erteilung noch die Laufzeit eines Patentes in irgendeiner Weise von einer derartigen Einstufung abhängig gemacht werden.

Begründung:

In der Praxis sind Patentansprüche auf Erfindungen im Ambivalenzbereich (siehe Abbildung) nicht eindeutig in ihren potentiellen Auswirkungen auf den Softwaremarkt zu beurteilen. Daher droht eine erhebliche Rechtsunsicherheit, wenn Rechtsfolgen an die Einordnung eines Patentes in die Kategorie der "Software-Patente" geknüpft wären.

★★ **PP−3:** Das künftige Patentrecht sollte eine kollektive Lizenzierung bzw. Rechtewahrnehmung ermöglichen.

Begründung:

Im Bereich der Wahrnehmung von Urheberrechten hat man seit langem erkannt, dass die an sich zustimmungspflichtigen Handlungen wie Vervielfältigung oder Verbreitung von Werkkopien in bestimmten Lebensbereichen, etwa im privaten Umfeld, nicht oder nur unter Inkaufnahme einer unverhältnismäßigen Gesetzesdurchsetzung kontrollierbar wären. Als Gegenmittel hat man in Gestalt der Verwertungsgesellschaften Instrumente zur kollektiven Lizenzierung von Urheberrechten geschaffen, die eine pauschalisierte Art der Vergütungsabwicklung durchführen und sowohl die Rechteinhaber als auch die Rechtenutzer davon entlasten, über jeden einzelnen Nutzungsvorgang einen individuellen Vertrag abschließen zu müssen.

Im Patentrecht gibt es bislang keine Parallele. Die Entscheidung darüber, ob eine Patentlizenz erforderlich ist, hängt heute in der Informationstechnik häufig nicht mehr von den Eigenschaften körperlicher Produkte, sondern von den Eigenschaften eines Computerprogrammes ab. Das Computerprogramm kann aber faktisch ohne nennenswerte Kosten − und im Fall von "Open Source"-Software auch rechtmäßig − in nahezu unbegrenzter Stückzahl vervielfältigt werden. Damit holt die Problematik der Rechtelizenzierung in ausgesprochenen Massenverfahren auch das Patentrecht ein.

★★ **PP−4:** Die Richtlinie des Rates über den Rechtsschutz von Computerprogrammen (91/250 EWG v. 14.5.1991) sollte geändert werden und das "Reverse Engineering" von Software, insbesondere die Dekompilation, generell zulassen.

Begründung:

Das derzeit geltende generelle Dekompilationsverbot ist schädlich.

- Es ist ein Anachronismus, wenn derjenige, der den Quellentext nicht offenlegt, mit einer zweifachen patentrechtlichen Privilegierung belohnt wird:

  → Zum einen gelten die in dem Code verwirklichten Ideen und Grundsätze als der Öffentlichkeit nicht zugänglich, wodurch eine Patentierung dieser Ideen und Grundsätze auch dann noch möglich erscheint, wenn der Objekt- oder Binärcode bereits allgemein erhältlich ist.

  → Zum anderen muss derjenige Patentinhaber, der durch Dekompilation eine Patentverletzung nachzuweisen sucht, mit umgehenden Gegenansprüchen des Verletzers aus dem urheberrechtlichen Dekompilationsverbot heraus rechnen.

- Das geltende Dekompilationsverbot verhindert eine wirksame Sicherheitsprüfung von als Binärcode vertriebener Software. Nur durch Reverse Engineering kann dem öffentlichen Interesse an IT-Sicherheit (siehe Empfehlung **IT−1**) geeignet Rechnung getragen werden.

★★ **PP−5:** Eine Neuheitsschonfrist von 12 Monaten soll in das Patentrecht (wieder) eingeführt werden.

Begründung:

Will ein Erfinder im Geltungsbereich des Europäischen Patentübereinkommens in den Genuss eines Patentschutzes für seine Erfindung kommen, ist er gezwungen, diese bis zur Einreichung der Patentanmeldung beim Patentamt gegenüber der Öffentlichkeit geheim zu halten.

Eine derartige Geheimhaltungspflicht ist mit Ethos und praktischer Arbeitsweise vieler "Open Source"-Softwareentwickler nicht vereinbar. Der Prozess der Softwareentwicklung wird im "Open Source"-Bereich regelmäßig in offenen, sich über das Internet konstituierenden Arbeitsgruppen geleistet. Jegliche Patentierbarkeit ist damit schon *in statu nascendi* − jedenfalls in Deutschland und Europa, nicht aber in den USA − vereitelt.

Wenn die Förderung von Patentaktivitäten der im "Open Source"-Bereich tätigen Akteure politisch für vorteilhaft gehalten wird, müssen den Entwicklern Möglichkeiten gegeben werden, ihre Erfindungen in patentverwertbarem Zustand zu behalten, ohne die Offenheit ihres Arbeitsmodus aufgeben zu müssen. Die Einführung einer Neuheitsschonfrist erscheint hierzu unabdingbar.

★★ **PP−6:** Das deutsche Patent- und Markenamt soll Offenlegungs- und Patentschriften unentgeltlich im Internet zum Abruf durch die Öffentlichkeit bereitstellen. Dasselbe gilt für Inhalte von Amtsakten und sonstige relevante Informationen.

Begründung:

Der Zugang zu Patentinformationen sollte über das Internet einfach, vollständig und kostenlos möglich sein:

- Die Patentanmelder haben mit den an das Patentamt entrichteten Gebühren bereits ein Entgelt für die Veröffentlichung ihrer Anmeldungen und Patentschriften entrichtet. Die unentgeltliche Publikation im Internet stellt unter diesen Umständen eine angemessene Gegenleistung dar.

- Durch den kostenlosen Zugang zu den Patentinformationen entfällt eine unzeitgemässe Hemmschwelle für die Information der Öffentlichkeit über den Stand der Technik. Kleine und mittlere Unternehmen wie auch "Open Source"-Entwickler würden nicht mehr durch zu hohe Informationsbeschaffungskosten benachteiligt.

★ **PP−7:** Es ist wünschenswert, eine Institution zu schaffen, die eine öffentliche digitale Zeitstempelstelle − im Sinne des Signaturgesetzes− für Quelltexte von "Open Source"-Software betreibt.

Begründung:

Das von der Zeitstempelstelle ausgefertigte, mit einer digitalen Signatur versehene Zertifikat kann von Einsprechenden oder Nichtigkeitsklägern später eingesetzt werden, um nachzuweisen, dass der Quellentext über einen gewissen Zeitraum hinweg öffentlich zugänglich war. Damit wird eine

infrastrukturelle Hilfe zur Verfügung gestellt, um schädliche Auswirkungen ungerechtfertigter Patente zu vermindern.

★ **PP−8:** Es ist wünschenswert, dass die Bundesregierung sich dafür einsetzt, dass das in Empfehlung **PP-2** geforderte **Quelltextprivileg** auch international eingeführt wird.

Begründung:

In Fällen, bei denen Computersoftware über das Internet verbreitet wird, bestehen schwierige Probleme der Rechtsanwendbarkeit, wenn das Herunterladen in einer Vielzahl von Ländern möglich ist, in denen Patente auf computer-implementierbare Erfindungen bestehen. Die breite internationale Einführung eines patentrechtlichen Quelltextprivilegs ist in Zeiten der Internetökonomie anzustreben, um national unterschiedliche Rechtsanwendungen im Interesse der Rechtssicherheit auszuschließen.

Durch Vereinbarungen auf der Ebene der WIPO, bei den anstehenden Beratungen im Rahmen des "WIPO Standing Committee on the Law of Patents" (SCP), wäre eine Lösung dieser Probleme im Prinzip erzielbar.

Empfehlungen zur wissenschaftlichen Bearbeitung des Themas

★ **WI−1:** Wünschenswert sind Untersuchungen zur Klärung der Wechselwirkungen von Patentschutz und Interoperabilität.

Begründung:

Im Urheberrecht für Computerprogramme ist das generelle Dekompilationsverbot in bestimmten Fällen durchbrochen worden, die der Herstellung von Interoperabilität* dienen. Dieser gewollte Effekt kann jedoch patentrechtlich vereitelt werden. Dadurch werden erhebliche Marktzutrittsbarrieren geschaffen, deren ökonomischer Wirkungsradius weit über den eigentlichen Patentschutzgegenstand hinausgeht.

Die schwierige Abwägung zwischen den Interessen des Patentinhabers und den Interessen der Softwareentwickler und Anwender an Interoperabilität setzt ein makroökonomisches Verständnis der Strukturen in entwickelten Informationsgesellschaften voraus, welches heute noch nicht zur Verfügung zu steht.

Möglicherweise würden die aus dem Urheberrecht bekannten Zwangslizenzen einen geeigneten Lösungsansatz bieten, das Interesse der Öffentlichkeit an Interoperabilität zu berücksichtigen. Das wäre zu prüfen.

★ **WI−2:** Dispute um die Patentpolitik leiden darunter, dass die jeweiligen Positionen fast immer empirisch nicht belegt sind. Es gibt auch keine ernst zu nehmenden Kennzahlen über die neuen Softwaremärkte. Es ist daher wünschenswert, insbesondere den Einfluss von Patenten auf die Softwareentwicklung und die Entwicklung der Volkswirtschaft als Ganzes wissenschaftlich zu untersuchen.

Begründung:

Über das Für und Wider von "Software-Patenten" sollte man nur auf der Basis gesicherter makroökonomischer Daten entscheiden. Dieser Gedanke ist in Deutschland − anders als in den USA − noch nicht weit verbreitet. Die unbefriedigende Datenlage zwingt dazu, sich in vielen Fällen auf amerikanische Studien zu beziehen.

Es wäre wünschenswert, Studien anzuregen, die die Probleme nicht einseitig − sei es aus juristischer, sei es aus bloß technischer, sei es aus rein ökonomischer Sicht − behandeln.

---

* Interoperabilität ist die Fähigkeit eines Programmes zum Austausch von Informationen und zur wechselseitigen Verwendung der ausgetauschten Informationen.

★ **WI−3:** Nach dem gegenwärtigen Wissensstand sprechen überwiegende Gründe für die Annahme, dass im Quelltext offengelegte Software die Sicherheit in der Informationstechnik entscheidend verbessert. Es ist wünschenswert, die von Informatikern behaupteten Zusammenhänge genauer zu untersuchen.

Begründung:

Die Betrachtung des Patentrechts aus der Sicht der IT-Sicherheit führt zu Fragestellungen, die praktisch noch nirgends untersucht sind.

Zu fördern sind insbesondere Studien aus folgenden Themengebieten:

- Durch welche Instrumentarien des Patentrechts kann ein Beitrag zur [marktwirtschaftlichen] Fortentwicklung des Datenschutzes erwartet werden? Denn Datenschutz moderner Prägung ist ohne ein hohes Niveau von IT-Sicherheit nicht mehr denkbar.

- Nach dem heutigen Stand der Kenntnisse sind wirkliche Erfolge im E-Commerce insbesondere davon abhängig, dass zuverlässige Kommunikationsformen geschaffen werden. Häufig werden diese mit digitalen Signaturen arbeiten. Es muss bezweifelt werden, ob Signaturverfahren, deren Source Code nicht offengelegt ist, das im Geschäftsverkehr notwendige Vertrauen schaffen werden.

Inhaltsverzeichnis

1. Software, Patentrecht, Urheberrecht: Was sind "Software−Patente"?

1.1. Allgemeines zum Patentrecht

Das Patentrecht umfasst drei besonders wichtige Gruppen von Vorschriften:

- Vorschriften, die die *Erlangung* eines Patentes durch den Patentanmelder regeln
- Vorschriften, die die *Wirkung* eines Patentes, d.h. dessen Durchsetzung durch den Patentinhaber gegenüber einem Patentverletzer, regeln
- Vorschriften, die die *Unterrichtung der Öffentlichkeit* hinsichtlich beantragter oder erteilter Patente regeln

Das Patentsystem

Patente können stets nur auf *Erfindungen* erteilt werden. *Cum grano salis* lässt sich sagen, dass in allen relevanten Rechtsordnungen Patente nur auf solche Erfindungen erteilt werden, die

- neu;
- nicht (völlig) trivial und
- gewerblich anwendbar sind.

Zur Erlangung eines Patentes sind erforderlich:

- ein Antrag auf Erteilung eines Patentes – die "Patentanmeldung" − bei einer zuständigen Zentralbehörde für den gewerblichen Rechtsschutz

und − darauf folgend −

- eine formelle Bestätigung der Behörde ("Patenterteilung").

Eine Patentanmeldung und die das daraufhin erteilte Patent wiedergebende Patentschrift enthalten typischerweise:

- eine Beschreibung der Erfindung, die so ausführlich gehalten sein muss, dass ein Fachmann sie nacharbeiten kann
- eine oder mehrere Zeichnungen, die dem Fachmann das Verständnis der Erfindung erleichtern sollen
- eine oder mehrere Patentansprüche, die genau bezeichnen, welche Merkmale die Erfindung umfasst

Ein Patent gibt seinem Inhaber das alleinige Recht, die patentierte Erfindung zu benutzen. Entscheidend hierfür sind die zum erteilten Patent gehörenden Patentansprüche. Jeder Patentanspruch besteht im Prinzip aus einem Gattungsbegriff, beispielsweise "Kraftfahrzeugbremse", "Verfahren zur Herstellung von Schwefelsäure" oder dergleichen, und einer Aufzählung von Merkmalen, die erfüllt sein müssen, damit ein unter den Gattungsbegriff fallender Gegenstand bzw. ein unter den Gattungsbegriff fallendes Verfahren als patentverletzend gilt. Die Frage, ob ein bestimmter lebensweltlicher Gegenstand oder ein bestimmtes lebensweltliches Verfahren ein Patent verletzt oder nicht, wird deshalb anhand eines Vergleiches der Merkmalsliste eines erteilten Patentanspruches mit den konkreten Eigenschaften des Gegenstandes oder Verfahrens bestimmt. Nur sofern *jedes* der Merkmale des Patentanspruches wörtlich oder seiner wesentlichen Wirkung nach erfüllt ist, liegt eine Patentverletzung vor.

Umgangssprachlich pflegt man zu sagen, dieser oder jener Gegenstand, etwa ein Kraftfahrzeugfahrwerk oder ein Verfahren zur Herstellung von Schwefelsäure, sei "patentiert". Diese Ausdrucksweise stellt jedoch eine Vergröberung dar, denn das Patent stellt – entgegen der Konnotation dieses Ausdruckes – keineswegs ein dingliches Recht dar, welches gewissermaßen dem "patentierten" Gegenstand anhaftet. Richtig ist vielmehr, dass das Patentrecht zu den Immaterialgüterrechten zählt und stets nur auf eine zielgerichtete Lehre zur Lösung einer bestimmten Aufgabe gerichtetet ist. Die Ursache für dieses Missverständnis dürfte darin liegen, dass die den Schutzbereich eines Patentes bestimmenden Patentansprüche stets von einem Gattungsbegriff ausgehen und diesen durch eine Aufzählung von Merkmalen gegenüber nicht unter das Patent fallenden Gegenständen abgrenzen.

1.2. Software als Gegenstand von Urheber- und Patentrecht

Ist Software nach derzeitiger Rechtslage ein legitimer Gegenstand des Urheberrechtes, des Patentrechtes oder beider Schutzrechtsarten?

Der Schutz von Software ist in Deutschland mit der Urheberrechtsnovelle von 1985 primär dem Urheberrecht zugewiesen worden. Dies darf jedoch nicht zu dem Fehlschluss führen, Datenverarbeitungsprogramme seien damit gewissermaßen per Definition aus dem Wirkungsbereich des Patentrechtes herausgenommen. Dies hat die Europäische Kommission im Frühjahr 2000 so formuliert:

> «Die Kommission hat festgestellt, daß bestimmte Kreise von der irrigen Annahme ausgehen, das Urheberrecht sei das einzige Mittel zum Schutz von Computerprogrammen. Daher soll auf Artikel 9 (1) 1. Satz der Richtlinie hingewiesen werden, nach dem sonstige Rechtsvorschriften, etwa das Patentrecht und andere Vorschriften zum Schutz von geistigem Eigentum, unberührt bleiben.»[16]

Umgekehrt darf die Rolle des Patentrechtes im Hinblick auf computer-implementierbare Erfindungen nicht isoliert gesehen werden, denn insbesondere bei der Regulierung des "Reverse Engineering" wirkt das Urheberrecht auf patentrechtliche Sachverhalte zurück.

---

16 **Bericht zur Software-Richtlinie (91/250/EWG) 2000.**

Nur eine Betrachtung des *gesamten* gegenwärtigen rechtlichen Umfeldes von Datenverarbeitungsprogrammen unter Einschluss von Patent- und Urheberrecht kann den anstehenden Fragestellungen gerecht werden.

Das Urheberrecht betrachtet ein Datenverarbeitungsprogramm aus einer linguistischen Perspektive. Das Datenverarbeitungsprogramm, der "Programmcode", wird als Aneinanderreihung von − aus einem bestimmten vorgegebenen Alphabet entnommenen − Zeichen nach Maßgabe der Grammatik einer formalen Sprache, der sogenannten *Programmiersprache*, aufgefasst. Diese linguistische Sicht eines Datenverarbeitungsprogrammes rechtfertigt es, das Datenverarbeitungsprogramm als Sprachwerk aufzufassen und – sofern es sich um eine eigene persönliche Schöpfung handelt – dem Programmierer als Urheber für die von ihm geschöpfte besondere sprachliche *Ausdrucksform* Urheberrechtsschutz zuzubilligen.

Demgegenüber hat das Patentrecht statt des linguistischen Aspektes die durch den Programmcode ausgedrückte *Funktionalität* im Blick. Es schützt nicht die vom Urheber geschöpfte sprachliche Ausdrucksform, sondern eine vom Erfinder geschaffene "Lehre", also eine Anweisung zu einem auf einen bestimmten Erfolg hin ausgerichteten Handeln.

Im Hinblick auf patentrechtliche Aspekte der Datenverarbeitungsprogramme kann der Programmcode nicht isoliert und für sich allein betrachtet werden. Er muss vielmehr im Zusammenhang mit dem "Prozessor" gesehen werden, auf dem er abläuft, denn nur so lässt sich die Funktionalität des Programmcodes objektiv bestimmen. Der Prozessor darf dabei nicht im engen Sinne als ein bestimmtes Stück Hardware, etwa eine Zentraleinheit eines Personalcomputers, aufgefasst werden; es kommt vielmehr auch eine "virtuelle Maschine" in Betracht, bei der eine bestimmte Prozessor--Hardware mit einer Interpreter- oder Compiler-Software zusammenwirkt, um ein linguistisches Konstrukt einer Programmiersprache, die nicht von der Hardware unmittelbar verarbeitet werden kann, dennoch abarbeiten zu können. Bei einer patentrechtlichen Betrachtungsweise kommt es also stets auf den "*Running Code*", nicht auf ein statisches linguistisches Konstrukt an.

Durch die Arbeit des Programmierers entsteht somit nicht nur das eigentliche Datenverarbeitungsprogramm als urheberrechtlich zu betrachtendes linguistisches Konstrukt. Da das Datenverarbeitungsprogramm stets auch dazu bestimmt ist, auf einer realen Hardware abzulaufen, formt der Programmierer implizit mit jedem von ihm geschöpften Programm auch eine patentrechtlich bedeutsame Vorrichtung, nämlich einen Computer. Dieser Computer wird durch die im Programm ausgedrückte Funktionalität gesteuert und zeigt nach außen ein bestimmtes Verhalten. Die zeitliche Aufeinanderfolge der Verarbeitungsschritte, die der Computer unter der Kontrolle des Programmes ausführt, stellt ein Verfahren dar, das patentrechtlich von Bedeutung sein kann. Beide Aspekte, sowohl die Eigenschaften der aus Hardware und Software entstehenden Vorrichtung als auch die Charakteristika der in dem Gesamtsystem aus Hardware und Software ablaufenden Verarbeitungsschritte, erhalten ihre Ausprägung durch die Software, ohne dass diese isoliert und für sich genommen bei der patentrechtlichen Beurteilung eine Rolle spielt. Das vom Computer gedanklich völlig losgelöste Datenverarbeitungsprogramm ist als reines linguistische Konstrukt

patentrechtlich stets bedeutungslos, denn seine Funktionalität erschließt sich erst aus der Wechselwirkung mit dem ihm zugedachten Prozessor.

Der urheberrechtliche und der patentrechtliche Schutz von Software zielen somit auf unterschiedliche Aspekte ein und desselben Lebenssachverhaltes, ohne sich zu überlappen. Das Urheberrecht reduziert das Datenverarbeitungsprogramm auf ein statisches linguistisches Konstrukt, wohingegen das Patenrecht die sich im Zusammenwirken mit einem "Prozessor" entwickelnde Funktionalität im Blick hat. Die urheberrechtlichen und patentrechtlichen Aspekte von Software sind in ihrer gegenseitigen Komplementarität somit zueinander "orthogonal".

1.3. Patentschutz für Computer−implementierbare Erfindungen

1.3.1. Computer-implementierte und Computer−implementierbare Erfindungen

Durch die Patentansprüche aus "Software-Patenten" wird keinesfalls − wie der Begriff zunächst nahezulegen scheint − ein Monopol auf eine "Software" als linguistischem Konstrukt beansprucht. Hierzu folgendes Beispiel:

Computerprogramm, bestehend aus folgender Befehlsfolge:

```
public class JasminVisitor implements Visitor, Constants {
  private JavaClass       clazz;
  private PrintWriter     out;
  private String          class_name;
  private ConstantPoolGen cp;
  public JasminVisitor(JavaClass clazz, OutputStream out) {
    this.clazz = clazz;
    this.out   = new PrintWriter(out);
    class_name = clazz.getClassName();
    cp = new ConstantPoolGen(clazz.getConstantPool());
  }
  /**
   * Start traversal using DefaultVisitor pattern.
   */
  public void disassemble() {
    new DefaultVisitor(clazz, this).visit();
    out.close();
  }
[...]
}
```

Derartige Patentansprüche auf Programmcode sind bislang nicht erteilt worden. Sie bilden auch nicht den Gegenstand der Auseinandersetzung um den Patentschutz für Software-bezogene Erfindungen. Die derzeitige Fragestellung befasst sich nämlich mit der Patentfähigkeit von Patentansprüchen, die einen algorithmenartigen Aufbau aufweisen und denen anhand ihrer Merkmale anzusehen ist, dass sie auf eine Realisierung mittels eines zweckdienlich programmierten Universalrechners abzielen. Es wäre daher wohl besser, von "*informationstechnischen Algorithmenpatenten*" zu sprechen.

Die Tendenz zur Verwissenschaftlichung des Erfindungsprozesses[17] einerseits und die zunehmende Implementation von unterschiedlichsten Funktionalitäten mit Mitteln der Datenverarbeitungstechnik andererseits führen dazu, dass sich der Erfindungsbesitz häufiger in Gestalt eines Datenverarbeitungsprogrammes manifestiert, wobei ein Mikroprozessor über zweckentsprechende Ein-/Ausgabe-Schnittstellen in physikalische Kausalketten eingebunden ist. Es verwundert daher wenig, dass die Ausdehnung der Praxis bei der Erteilung von Patenten auf Software-Erfindungen bei den "Embedded Systems" sozusagen "durch die Hintertür" begonnen worden ist. Spätestens mit dem Aufkommen der Internet-Technologie scheint diese Begrenzung von Patenten auf Software-Erfindungen nicht mehr haltbar zu sein.

Diese Verwissenschaftlichung des Erfindungsprozesses hat erhebliche Konsequenzen für den Alltag des Patentwesens: Man kann heute die im Patentanspruch umrissene Erfindung nicht mehr bloß als eine Kombination konkreter Konstruktionsmerkmale auffassen, die aus einer Vielzahl potentiell möglicher und dem Erfinder gegenwärtiger technischer Realisierungsoptionen willkürlich herausgegriffen wurde.

Beispiel: "Antiblockiersystem"

Ein neu erfundenes Antiblockiersystem ("ABS") einer Kraftfahrzeugbremse sei in einem ersten Labormodell rein elektromechanisch realisiert worden. Eine darauf gerichtete Patentanmeldung wird sich unter den heutigen Rahmenbedingungen des verwissenschaftlichten Entwicklungsbetriebes nicht damit begnügen, die Zusammenfügung der Einzelteile dieses Labormodells zu beanspruchen. Was die Entwicklungsingenieure hervorgebracht haben, ist mehr als das Labormodell, nämlich ein abstraktes Modell einer ABS-Bremseinrichtung *als System miteinander in Wechselwirkung stehender Funktionskomponenten*. Wie diese Funktionskomponenten schließlich konkret realisiert werden, etwa hydraulisch, elektrohydraulisch oder auf andere Weise, bleibt offen. Dort, wo eine Funktionskomponente elektrische Signale verarbeitet, kann ein mit funktionalen Merkmalsangaben abgefasster Patentanspruch auch eine Software-Lösung mit einem Mikroprozessor umfassen.

Diese Entwicklung hat im Effekt dazu geführt, dass breite *funktionale Merkmalsformulierungen* in Patentansprüchen heute eher die Regel denn die Ausnahme sind. Bei funktionalen Merkmalen kann insbesondere im Falle von Patentgegenständen im Bereich der Signalverarbeitung häufig offen bleiben, ob diese bei (potentiellen) Verletzungsgegenständen durch eine konventionelle monofunktionale elektronische Schaltungsanordnung oder durch einen in geeigneter Weise programmierten Universalrechner realisiert werden. Funktionale Merkmalsformulierungen in Patentansprüchen bilden die heute gegebenen technischen Realitäten verwissenschaftlichter industrieller Innovationsprozesse ab.

Dieser heute vorherrschende Innovationsprozess unterscheidet sich von dem, was der typische Erfinder als "Tüftler" im XIX. Jahrhundert getrieben haben mag, durch ein hohes Abstraktionsniveau.

17 Vgl. **HORNS 2000.**

Festzuhalten bleibt, dass dieses Abstraktionsniveau genuin zu sein scheint, d.h. typische Wissenschaftler-Erfinder entwickeln ihre innovativen Ideen auf einer möglichst hohen Abstraktionsstufe. Es gibt keine Anzeichen dafür, dass die Abstraktionsleistung erst wesentlich nachträglich im Patentierungsverfahren eingebracht wird. Dies schließt nicht aus, dass beispiellsweise im Prozess des Abfassens einer Patentanmeldung weitere Abstrahierungspotentiale aufgedeckt werden.

Es versteht sich, dass andere Patentgegenstände bestehen, die entweder insbesondere keinerlei Bezug auf irgendeine Art von Signalverarbeitung nehmen − und bei denen somit kaum vorstellbar ist, dass sie sich auf den Software-Markt auswirken könnten − oder die *expressis verbis* auf Begriffe der Informationstechnik Bezug nehmen und so von sich aus klarstellen, dass mit ihnen eine Einflussnahme auch auf den Software-Markt beabsichtigt ist. Bei derartigen Fällen treten insoweit keine besonderen Beurteilungsprobleme auf.

Daraus folgt die Notwendigkeit einer Differenzierung zwischen zwei Bereichen von Erfindungen:

1. Erfindungen, die prinzipiell überhaupt nicht durch einen Computer mit darauf ablaufendem Datenverarbeitungsprogramm verkörpert werden können, wie etwa einer erfinderisch verbesserten rein mechanisch wirkenden Pleuelstange oder einer neuen chemischen Substanz,

und

2. Erfindungen, die im Prinzip durch einen Computer mit darauf ablaufendem Datenverarbeitungsprogramm verkörpert werden können, etwa einem kryptographischen Chiffriergerät, einem Telefon, einem MP3-Player oder einem Internet-Online-Einkaufssystem.

Der Begriff der "Erfindung" ist bei dieser Unterscheidung stets im Sinne von "Gegenstand eines Patentanspruches" aufzufassen.

Damit wird deutlich, dass diese beiden Bereiche von Erfindungen keinesfalls disjunkte Mengen darstellen. Vielmehr weisen sie einen ausgeprägten Überlappungsbereich ("Ambivalenzbereich") auf, dem solche Erfindungen zuzuordnen sind, die bei Bedarf sowohl mit Software als auch ohne Software realisierbar sind (siehe die Abbildung im Abschnitt "Empfehlungen an die Politik"). Insbesondere auf Signalverarbeitung zielende funktionale Merkmalsangaben in Patentansprüchen begründen den Ambivalenzbereich. Hierzu folgendes Beispiel:

Beispiel: IDEA-Verschlüsselungsalgorithmus (EP 0 482 154 B1 vom 30.06.1993)[18]

«[...] Vorrichtung für das Umwandeln jeweils eines beliebigen ersten binären Digitalblockes einer ersten Länge in einen zugeordneten zweiten binären Digitalblock gleicher Länge unter Verwendung von wenigstens einem frei wählbaren binären Steuerblock, gekennzeichnet

- durch wenigstens einen ersten Eingang zum Eingeben von wenigstens zwei ersten Teilblöcken einer zweiten Länge, die zusammen den ersten Digitalblock bilden,
- durch wenigstens einen zweiten Eingang zum Eingeben von wenigstens zwei Steuerblöcken der zweiten Länge,
- durch eine Logik, die jeweils nacheinander wenigstens vier logische Operationen wenigstens zweier unterschiedlicher Sorten durchführt, wobei wenigstens die überwiegende Zahl aller Paare unmittelbar aufeinanderfolgender Operationen aus zwei Operationen unterschiedlicher Sorten besteht, wobei durch jede Operation jeweils zwei Eingangsblöcke der zweiten Länge in einen Ausgangsblock dieser Länge umgewandelt werden, wobei als Eingangsblöcke erste Teilblöcke, Steuerblöcke und / oder Ausgangsblöcke einer jeweils vorhergehenden Operation dienen, und
- durch wenigstens einen Ausgang zum Ausgeben von wenigstens zwei den ersten Teilblöcken zugeordneten zweiten Teilblöcken der zweiten Länge, die zusammen den zweiten Digitalblock bilden.»

Dem Wortlaut des Patentanspruches ist keinerlei Hinweis auf eine Implementation durch Software zu entnehmen. Die wirtschaftliche Bedeutung dieses Patentanspruches liegt bei angemessener Würdigung aller Umstände jedoch ohne Zweifel darin, auch solche Verschlüsselungssysteme abzudecken, bei denen der IDEA-Verschlüsselungsalgorithmus in einer Software wie etwa PGP implementiert ist. Dessenungeachtet ist eine Hardware-Implementation nicht nur theoretisch möglich, sondern wird von der Patentinhaberin konkret in Gestalt eines Coprozessors vermarktet.

Es existiert somit eine mächtige Klasse von Zwitterobjekten, denen eine Relevanz hinsichtlich des Software-Marktes nicht einfach anzusehen ist. Dabei reicht es nicht aus, die Merkmale der Gegenstände von Patentansprüchen rein formal auf ihren Software-Bezug hin zu betrachten, wie im patentamtlichen Erteilungsverfahren üblich. Nur eine komplexe wertende Betrachtung, die auch außerhalb des Patentrechtes liegende Sachverhalte, etwa Kenntnisse über bestimmte Märkte, einbezieht, könnte zu einem Urteil führen, ob ein im Ambivalenzbereich begehrtes Patent einen wesentlichen Einfluss auf Software-Märkte auszuüben vermag oder nicht. Diese Aufgabe würde das eher kursorische patentamtliche Prüfungsverfahren strukturell überfordern.

Aufgrund dieser gravierenden Abgrenzungsproblematik zwischen "gewöhnlichen Patenten" und "Software-Patenten" steht auch keine genaue Statistik über den prozentualen Anteil beider Teilgruppen an der Gesamtzahl zur Verfügung. Zwar wird jede Patentanmeldung unmittelbar nach der Einreichung vom Patentamt nach der "Internationalen Patentklassifikation" einem technischen

---

18 Es handelt sich um das Patent der Ascom Tech AG Bern auf den sog. "IDEA"-Verschlüsselungsalgorithmus, der unter anderem in der Verschlüsselungssoftware "PGP" (Pretty Good Privacy) Verwendung findet. Im Internet: http://www.it-sec.ch/idea.html (2.9.2000).

Sachgebiet und damit auch einem für das Sachgebiet zuständigen Patentprüfer zugeordnet. Die Patentklassifikation erlaubt jedoch keine Differenzierung nach Software-Bezogenheit von Erfindungen im Ambivalenzbereich.

1.3.2. Die sogenannte "Technizitätsdebatte"

Die Abgrenzungsschwierigkeiten zwischen "gewöhnlichen" Patenten und "Software-Patenten" sind allgemein anerkannt. Man hat deshalb versucht, eine Trennung über den Begriff der "Technizität" zu erreichen.

Jede Software ist eine (programmier-)sprachliche Fixierung (Formalisierung) einer festgelegten Funktionalität. Software fällt in indirekter Weise und in genau dem Maße unter die Wirkung des Patentrechtes, in dem jene sprachlich fixierte Funktionalität als legitimer Gegenstand patentrechtlicher Ansprüche angesehen wird.

Sui−generis−Schutz

Ein etwaiges Sonderschutzrecht für Software würde bei der derzeitigen rechtlichen Ausgangslage lediglich den Eingriffspunkt des Patentrechtes auf Datenverarbeitungsprogramme überlagern. Die bestehenden Probleme würden dadurch nicht vermindert, sondern vermutlich sogar verstärkt. Versuche, die durch "Software-Patente" entstehenden Probleme (insbesondere im Umkreis der "Open Source"-Software) durch Einführung eines neuen *sui-generis*-Schutzes für Datenverarbeitungsprogramme außerhalb des komplementären Dualismus von Urheber- und Patentrecht zu lösen, müssten sich mit der Frage auseinandersetzen, auf welche gesetzestechnische Art und Weise die aus Hardware plus Software bestehenden Computersysteme dem Zugriff patentrechtlicher Ansprüche entzogen werden können.

Hierzu sind bislang noch keine überzeugenden Ansätze erkennbar.

"Technisch" und "untechnisch" als Abgrenzungskriterien

Ein Teil der Literatur wählt einen Lösungszugang über den gewohnheitsrechtlich ausgeformten Begriff der "Technizität" als notwendigem Attribut eines Patentgegenstandes.

Funktionalität, die von Hardware, gesteuert durch Software, herbeigeführt wird, sei per se "untechnisch" und somit dem Patentschutz nicht zugänglich, wenn die Erfindung in der Software implementiert ist.

Diese Position diskriminiert also bestimmte Funktionalitäten als "untechnisch" und sucht diese so aus der Reichweite des Patentrechts zu entfernen.

Die traditionelle Rechtsprechung hat "Technizität" de facto mit einer zielgerichteten Anwendung von Erkenntnissen der angewandten Physik gleichgesetzt. Strittig ist, ob der Gegenstand eines Patentanspruches bereits dann als "technisch" angesehen werden darf, wenn bereits ein einziges seiner

Merkmale, sei es für sich genommen neu oder bereits aus dem Stand der Technik bekannt, technisch ist.

Stellt man sich auf diesen Standpunkt, so sind computer-implementierbare Erfindungen stets technisch. Die Hardware, auf der die Software letztendlich abläuft, demonstriert schon durch ihren betriebsbedingten Energieumsatz, dass sie eine ingenieurtechnische Umsetzung physikalischer Erkenntnis verkörpert. Lediglich Erfindungen wie Geschäftsmethoden ohne jede technische Infrastrukur wären dann "untechnisch". Im Zusammenhang mit den Auswirkungen des Patentrechtes auf "Open Source"-Software spielt dieser Sonderfall jedoch insoweit keine Rolle, als bei der Programmausführung *per definitionem* stets ein Computer zugegen ist.

Weitere Differenzierung der Technizität

Gegenpositionen verlangen eine weitere Differenzierung und unterschiedliche Gewichtung einzelner Merkmale eines Patentanspruches im Hinblick auf deren "Technizität", um "Software-Erfindungen" aus dem Patentrecht möglichst weitgehend herauszuhalten.

Diese Ansätze erweisen sich aus mehreren Gründen als nicht tragfähig:

- Es kann nicht damit gerechnet werden, dass der Ambivalenzbereich dadurch quantitativ marginalisiert werden kann. Insbesondere im Bereich der "Embedded Systems" schaffen Programmierer oft Software, die zur Steuerung von Prozessoren dient, welche eine im patentrechtlichen Sinne "neue" Peripheriebeschaltung ansteuern. In derartigen Fällen wäre auch mit einer Technizitätsdiskriminierung von Patentanspruchsmerkmalen nicht zu verhindern, dass durch Software ein patentverletzender Gegenstand entsteht.
- Der Ambivalenzbereich wird dann zu einem "Flickenteppich" aus "technischen" beziehungsweise "nichttechnischen" Erfindungen. Das ist unter Gesichtspunkten der Rechtssicherheit in der Praxis nur schwer hinnehmbar. Die Relevanz des Patentrechtes für computer-implementierbare Erfindungen wird nicht aufgehoben.
- Dieser Ansatz schafft falsche Anreize für die Wirtschaft. Sie wird dazu neigen, statt flexibler, frei programmierbarer Mikroprozessorsysteme festverdrahtete Schaltungen zu verwenden, um den eigenen Wissensvorsprung patentrechtlich schützen zu können.
- Die "Technizitätsdebatte" ignoriert den Charakter der Hardware als physikalisches und darum technisches System. Sie trachtet danach, den Mikroprozessor ohne zureichende Begründung aus der Menge aller elektronischen Schaltungen herauszugreifen und den unter seiner Verwendung hervorgebrachten Lösungen einen minderen patentrechtlichen Status zuzuweisen. Wegen ihrer begrifflichen Inkonsistenzen weist die "Technizitätsdebatte" keinen Weg zur Lösung der anstehenden Probleme.

1.4. Wichtige und nicht so wichtige Fragen künftiger Patentpolitik

Die bisherige öffentliche Diskussion über das Für und Wider von Patenten auf computer--implementierbare Erfindungen leidet somit in weiten Bereichen an einer Fixierung auf den Bereich, der die Kriterien bei der *Erteilung* von Patenten betrifft. Infolge der durch den Ambivalenzbereich bedingten Unschärfe des Begriffes des "Software-Patentes" ist der Gestaltungsspielraum hier jedoch eher gering.

Man könnte natürlich auf den Gedanken kommen, lediglich die Erteilung von Patenten auf computer-implementierte Erfindungen zu unterbinden. Auch dieser Ansatz greift zu kurz. Denn die mit der Erstellung und dem Vertrieb von Software befassten Akteure müssten dann gegebenenfalls noch mit Verletzungen von Patenten aus dem Ambivalenzbereich rechnen.

Es sind derzeit keine zureichenden Gründe erkennbar, die es gerechtfertigt erscheinen lassen, den gesamten Bereich der computer-implementierbaren Erfindungen vom Patentschutz auszuschließen.

Eine Zurückführung des Bereiches der dem Patentschutz zugänglichen Gegenstände ausschließlich auf den Bereich der nicht computer-implementierbaren Gegenstände würde in der Praxis eine *radikale Beschneidung des Patentwesens* bedeuten, bei der – abgesehen von der Biotechnologie mit den ihr eigenen politischen Kontroversen – im Wesentlichen nur noch "klassische" Technikbereiche im Umkreis der "Old Economy" patentfähig wären. Es müssten funktionale Merkmalsformulierungen in Patentansprüchen ausgeschlossen werden − eine Einschränkung, durch die das Patentwesen dem verwissenschaftlichten Innovationsprozess nicht mehr gerecht werden könnte. Ohne den Ausschluss funktionaler Merkmalsformulierungen bestünde weiterhin die Möglichkeit, dass ein erteilter Patentanspruch auch Software-Lösungen auf einem Mikroprozessor umfasste.

Wie *Hellfeld*[19] bereits im Jahre 1989 angemerkt hatte, verliert die physische Konstitution der althergebrachten Patentgegenstände mehr und mehr ihre angestammte Bedeutung:

> «Aus der Maschinenentwicklung der letzten Jahrzehnte folgt zwingend die Einsicht, daß das Wesen der Maschine nicht in ihrer Körperlichkeit besteht. Die alte Maschine ist nur ein Sonderfall des neuen, viel umfassenderen Maschinentyps. In der deutschen Kulturtradition mit Berührungsängsten vor naturwissenschaftlichen und technischen Entwicklungen herrscht heute noch weithin ein klassisch-mechanischer Maschinenbegriff vor, wie man ihn im Physikunterricht der Schule gelernt hat. [...] Entscheidend für das, was der Computer als Maschine leistet, ist nicht so sehr seine materielle Ausrüstung, die Hardware, also seine körperliche Erscheinung, sondern eben sein Programm, die Software, etwas völlig Immaterielles.»

Es ist also geboten, sich eingehend mit den Vorschriften des Patentrechts zu befassen, die:

- die *Wirkung* eines Patentes regeln;
- die *Unterrichtung der Öffentlichkeit* hinsichtlich beantragter oder erteilter Patente regeln.

---

19 **Hellfeld 1989**.

Es ist also weniger wichtig, *ob* "Software-Patente" erteilt werden. Dies ist eine nur noch historisch interessante Frage. Diese Auffassung entspricht auch den Einsichten der neueren amerikanischen Literatur.[20]

2. Das Patentrecht in wichtigen Wirtschaftsräumen

2.1. Bundesrepublik Deutschland

2.1.1. Allgemeines zum Patentwesen in Deutschland

Ein Patent mit Wirkung für die Bundesrepublik Deutschland kann erteilt werden durch

- das Deutsche Patent- und Markenamt (DPMA) in München auf der Grundlage des Deutschen Patentgesetzes
- das Europäische Patentamt (EPA) in München auf der Grundlage des Europäischen Patentübereinkommens (EPÜ, auch "Münchner Übereinkommen")

Die *Wirkung* sowohl eines vom DPMA als auch eines vom EPA für Deutschland erteilten Patentes sind gleich und richten sich ausschließlich nach dem Patentgesetz (PatG). Im Folgenden sollen zunächst einige Rahmenbedingungen der Patenterteilung durch das DPMA näher betrachtet werden.

2.1.2. Patentfähige Gegenstände

Die Definition der nach nationalem deutschem Patentrecht patentfähigen Gegenstände findet sich in **§§1, 2 PatG**.

§§1, 2 PatG

«§ 1 [Erteilungsvoraussetzungen]

(1) Patente werden für Erfindungen erteilt, die neu sind, auf einer erfinderischen Tätigkeit beruhen und gewerblich anwendbar sind.

(2) Als Erfindungen im Sinne des Absatzes 1 werden insbesondere nicht angesehen:

1. Entdeckungen sowie wissenschaftliche Theorien und mathematische Methoden;
2. ästhetische Formschöpfungen;
3. Pläne, Regeln und Verfahren für gedankliche Tätigkeiten, für Spiele oder für geschäftliche Tätigkeiten sowie Programme für Datenverarbeitungsanlagen;
4. die Wiedergabe von Informationen.

(3) Absatz 2 steht der Patentfähigkeit nur insoweit entgegen, als für die genannten Gegenstände oder Tätigkeiten als solche Schutz begehrt wird.

20 Vgl. **Cohen/Lemley 2000 (2001)**.

§ 2 [Keine Erteilung]

Patente werden nicht erteilt für

1. Erfindungen, deren Veröffentlichung oder Verwertung gegen die öffentliche Ordnung oder die guten Sitten verstoßen würde; ein solcher Verstoß kann nicht allein aus der Tatsache hergeleitet werden, daß die Verwertung der Erfindung durch Gesetz oder Verwaltungsvorschrift verboten ist. Satz 1 schließt die Erteilung eines Patents für eine unter § 50 Abs. 1 fallende Erfindung nicht aus.
2. Pflanzensorten oder Tierarten sowie für im wesentlichen biologische Verfahren zur Züchtung von Pflanzen oder Tieren. Diese Vorschrift ist nicht anzuwenden auf mikrobiologische Verfahren und auf die mit Hilfe dieser Verfahren gewonnenen Erzeugnisse.»

Programme für Datenverarbeitungsanlagen "*als solche*" sind vom Patentschutz ausgeschlossen.

Ein Erfordernis, wonach ein patentfähiger Gegenstand "technisch" zu sein hat, ist im deutschen Patentgesetz nicht *expressis verbis* enthalten. Diese "Technizität" wird aber von der Rechtsprechung gefordert.

2.1.3. Rechtsprechung

Es gibt in der Rechtsprechung des Bundesgerichtshofs (BGH)[21] ein ungebrochenes Bekenntnis zum – bislang im PatG noch ungeschriebenen – Erfordernis der *Technizität* des Patentgegenstandes. Der BGH hat im Jahre 1969 in der Entscheidung **BGH "Rote Taube" (1969)** diesen Begriff wie folgt definiert:

> «Technisch ist eine Lehre zum planmäßigen Handeln unter Einsatz beherrschbarer Naturkräfte zur Erreichung eines kausal übersehbaren Erfolges, der ohne Zwischenschaltung menschlicher Verstandestätigkeit die unmittelbare Folge des Einsatzes beherrschbarer Naturkräfte ist.»

Eine technische Lehre ist demnach also eine planmäßige, zielgerichtete Instrumentalisierung der angewandten Physik.

Die derzeitige Praxis ist von einem allmählichen Wandlungsprozess der Anschauungen hinsichtlich der patentrechtlichen Bedeutung der physikalischen Abläufe in der Zentraleinheit eines ein Programm abarbeitenden Rechners bestimmt. Er führt dazu, dass im Gegenstand, der durch den Patentanspruch bestimmt wird, ein Wirken von Naturkräften auch dann unterstellt wird, wenn auch −

---

21 Einen aktuellen Überblick über die deutsche Rechtsprechung gibt − in englischer Sprache − **Bechtold 2000**.

im Minimalfall − *nur die Zentraleinheit* als physikalisches System identifizierbar ist (**BGH "Sprachanalyseeinrichtung" (2000)**).

Als historischer Ausgangspunkt für diesen Wandlungsprozess bei der Beurteilung von Patenten auf software-bezogene Erfindungen durch den BGH kann die "Kerntheorie" gelten.[22] Rechenprogramme für elektronische Datenverarbeitungsanlagen, bei deren Anwendung lediglich von einer in Aufbau und Konstruktion bekannten Datenverarbeitungsanlage bestimmungsgemäßer Gebrauch gemacht werde, sind dieser − inzwischen aufgegebenen − Auffassung nach auch dann *nicht patentfähig*, wenn mit Hilfe der Datenverarbeitungsanlage ein Herstellungs- oder Bearbeitungsvorgang mit bekannten Steuerungsmitteln unmittelbar beeinflusst wird. Es komme darauf an, in welchen Anweisungen der als neu und erfinderisch beanspruchte Kern der Lehre zu sehen sei, d.h. in welchen Schritten das Problem der fertigen Lösung zugeführt werde.

Im Jahre 1992 hat der BGH mit der Entscheidung **BGH "Tauchcomputer" (1992)** die Kerntheorie praktisch aufgegeben. Bei Gegegenständen, die technische und nichttechnische Merkmale verknüpfen, muss der gesamte Gegenstand unter Einschluss der etwaigen Rechenregel berücksichtigt werden. Es dürfe nicht der Gegenstand zerlegt und dann nur der Teil auf Naheliegen geprüft werden, der aus den technischen Merkmalen besteht.

In jüngster Zeit nimmt der BGH in der Entscheidung **BGH "Logikverifikation" (2000)** Abschied von der Vorstellung, patentfähige Software-Erfindungen müssten stets wirkungsmäßig unmittelbar in physikalische Kausalketten eingebunden sein. Er stellt nunmehr fest, dass die Frage, ob eine auf ein Programm für Datenverarbeitungsanlagen gerichtete Patentanmeldung die erforderliche Technizität aufweise, eine wertende Betrachtung des im Patentanspruch definierten Gegenstandes erfordere. Es soll nämlich im Einzelfall genügen, wenn der Patentgegenstand einen Lösungsvorschlag in einem Prozess betrifft, der auf den unmittelbaren Einsatz von beherrschbaren Naturkräften verzichtet und die Möglichkeit der Fertigung tauglicher Erzeugnisse anderweitig durch auf technischen Überlegungen beruhende Erkenntnisse voranzubringen sucht. Zwischen den im beanspruchten Algorithmus angegebenen Begriffen und der Physik der Naturkräfte muss es somit noch ein schmales Band «*auf technischen Überlegungen beruhender Erkenntnisse*» geben, welches für die Patentfähigkeit ausschlaggebend sein soll.

In der späteren Entscheidung **BGH "Sprachanalyseeinrichtung" (2000)** stellt das Gericht fest, dass einer Datenverarbeitungsanlage als Vorrichtung, die in bestimmter Weise programmtechnisch eingerichtet ist, technischer Charakter zukommt. Dabei komme es für die Beurteilung des technischen Charakters einer solchen Vorrichtung nicht darauf an, ob mit ihr ein (weiterer) technischer Effekt erzielt wird, ob die Technik durch sie bereichert wird oder ob sie einen Beitrag zum Stand der Technik leistet. Auch stehe dem technischen Charakter der Vorrichtung nicht entgegen, dass ein Eingreifen des Menschen in den Ablauf des auf dem Rechner durchzuführenden Programmes in Betracht komme.

---

22 **BGH "Dispositionsprogramm" (1977).**

Man darf dieses Diktum des BGH wohl dahingehend interpretieren, dass alle Anspruchsgegenstände als technisch angesehen werden, bei denen auch nur ein einziges Merkmal direkt eine physikalische Interaktion beinhaltet oder – wie in der Entscheidung **BGH "Logikverifikation" (2000)** – indirekt mit physikalischen Interaktionen mindestens begrifflich in Zusammenhang steht. Im Zweifel reicht der Energieumsatz der Zentraleinheit, um die erforderliche Technizität herzustellen.

2.2. Das Europäische Patentübereinkommen (EPÜ)

2.2.1. Die Funktionsweise des EPÜ

Das EPÜ begründet eine *Europäische Patentorganisation* (EPO), die das *Europäische Patentamt* (EPA) betreibt. Die EPO steht als solche zunächst außerhalb der Europäischen Union und ihrer Organe oder Unterorganisationen Sie stellt vielmehr ein eigenständiges Völkerrechtssubjekt dar. Dem EPÜ gehören auch Staaten an, die nicht Mitglied der EU sind. Das Europäische Patentamt ist kraft der Bestimmungen des EPÜ befugt, für die Mitgliedsstaaten Patente zu erteilen, die die gleiche Wirkung wie national erteilte Patente aufweisen.

2.2.2. Patentfähige Gegenstände nach dem EPÜ

Die Abgrenzung der patentfähigen Gegenstände findet sich in den **Artikeln 52 u. 53 EPÜ**:

Artikel 52, 53 EPÜ

«Artikel 52 - Patentfähige Erfindungen

(1) Europäische Patente werden für Erfindungen erteilt, die neu sind, auf einer erfinderischen Tätigkeit beruhen und gewerblich anwendbar sind.

(2) Als Erfindungen im Sinn des Absatzes 1 werden insbesondere nicht angesehen:

  a) Entdeckungen sowie wissenschaftliche Theorien und mathematische Methoden;

  b) ästhetische Formschöpfungen;

  c) Pläne, Regeln und Verfahren für gedankliche Tätigkeiten, für Spiele oder für geschäftliche Tätigkeiten sowie Programme für Datenverarbeitungsanlagen;

  d) die Wiedergabe von Informationen.

(3) Absatz 2 steht der Patentfähigkeit der in dieser Vorschrift genannten Gegenstände oder Tätigkeiten nur insoweit entgegen, als sich die europäische Patentanmeldung oder das europäische Patent auf die genannten Gegenstände oder Tätigkeiten als solche bezieht.

(4) Verfahren zur chirurgischen oder therapeutischen Behandlung des menschlichen oder tierischen Körpers und Diagnostizierverfahren, die am menschlichen oder tierischen Körper vorgenommen werden, gelten nicht als gewerblich anwendbare Erfindungen im Sinn des Absatzes 1. Dies gilt nicht für Erzeugnisse, insbesondere Stoffe oder Stoffgemische, zur Anwendung in einem der vorstehend genannten Verfahren.

Artikel 53 - Ausnahmen von der Patentierbarkeit

Europäische Patente werden nicht erteilt für:

a) Erfindungen, deren Veröffentlichung oder Verwertung gegen die öffentliche Ordnung oder die guten Sitten verstoßen würde; ein solcher Verstoß kann nicht allein aus der Tatsache hergeleitet werden, daß die Verwertung der Erfindung in allen oder einem Teil der Vertragsstaaten durch Gesetz oder Verwaltungsvorschrift verboten ist;

b) Pflanzensorten oder Tierarten sowie für im wesentlichen biologische Verfahren zur Züchtung von Pflanzen oder Tieren; diese Vorschrift ist auf mikrobiologische Verfahren und auf die mit Hilfe dieser Verfahren gewonnenen Erzeugnisse nicht anzuwenden.»

Ebenso wie im nationalen deutschen Patentrecht sind "Programme für Datenverarbeitungsanlagen als solche" vom Patentschutz ausgenommen. Auch das Erfordernis der "Technizität" ist bislang nicht *expressis verbis* in das Abkommen aufgenommen worden, wird jedoch von der Rechtsprechung der Beschwerdekammern[23] des Europäischen Patentamtes gefordert.

Aus den Prüfungsrichtlinien des Europäischen Patentamtes ergeben sich weitere Anhaltspunkte für die gegenwärtige Amtspraxis (s. Anhang C).[24]

2.2.3. Rechtsprechung der Beschwerdekammern des Europäischen Patentamtes (EPA)

Ein im Vergleich zur Rechtsprechung des BGH ähnlicher Wandlungsprozess ist hinsichtlich der Anschauungen der Beschwerdekammern des EPA eingetreten.

In der Entscheidung **"Computerprogrammprodukt / IBM"**[25] hält eine *Beschwerdekammer des EPA* jedoch daran fest, dass davon ausgegangen wird, dass Computerprogrammen nicht allein deshalb ein technischer Charakter zugesprochen werden könne, weil sie Computerprogramme seien. Dies bedeute, dass die bei Ausführung von Programmbefehlen auftretenden physikalischen Veränderungen bei der Hardware (die beispielsweise elektrische Ströme fließen lassen) nicht per se den technischen Charakter ausmachen könnten, durch den das Patentierungsverbot für ein solches Programm gegenstandslos werden würde. Solche Veränderungen könnten zwar als etwas Technisches angesehen werden, wären aber ein gemeinsames Merkmal aller auf einem Computer lauffähigen Programme und eigneten sich daher nicht zur Unterscheidung von Computerprogrammen mit technischem Charakter einerseits und Computerprogrammen "als solchen" andererseits. Daher müsse

---

23 Die Beschwerdekammern des Europäischen Patentamtes sind zwar organisatorisch in das Amt eingebunden, sollen aber im Hinblick auf ihre sachliche Unabhängigkeit und Weisungsungebundenheit eine gerichtsähnliche Funktion bei der Überprüfung von Beschlüssen des Amtes ausüben.

24 Bei den Prüfungsrichtlinien handelt es sich um interne Verwaltungsanweisungen des EPA, die nicht den Charakter einer eigenständigen Rechtsquelle aufweisen, die jedoch einen gleichmäßigen Verwaltungsvollzug des EPÜ sicherstellen sollen.

25 **Technische Beschwerdekammer d. EPA "Computerprogrammprodukt / IBM" (1998)**.

anderswo nach einem technischen Charakter gesucht werden: Er könne in den weiteren Effekten liegen, die mit der Ausführung der Programmbefehle einhergehen.

In einer (unveröffentlichten) Entscheidung (T 0935/977 vom 4. Februar 1999) äußert sich dieselbe *Beschwerdekammer des EPA* wie folgt:

> «[...] For the purpose of interpreting the exclusion from patentability of programs for computers under Article 52(2) and (3) EPC, it is assumed that programs for computers cannot be considered as having a technical character for the very reason that they are programs for computers. [...] This means that physical modifications of the hardware (causing, for instance, electrical currents) deriving from the execution of the instructions given by programs for computers cannot per se constitute the technical character required for avoiding the exclusion of those programs. [...] Although such modifications may be considered to be technical, they are a common feature of all those programs for computers which have been made suitable for being run on a computer, and therefore cannot be used to distinguish programs for computers with a technical character from programs for computers as such. [...] It is thus necessary to look elsewhere for technical character in the above sense: It could be found in the further effects deriving from the execution (by the hardware) of the instructions given by the computer program. Where said further effects have a technical character or where they cause the software to solve a technical problem, an invention which brings about such an effect may be considered an invention, which can, in principle, be the subject-matter of a patent. [...]»

In den Leitsätzen der jüngst ergangenen Entscheidung **"Controlling pension benefits system / PBS PARTNERSHIP"**[26] grenzt eine Beschwerdekammer des EPA die Rolle nichttechnischer Merkmale ökonomisch-geschäftlicher Natur in Patentansprüchen ab:

---

26 **Technische Beschwerdekammer d. EPA "Controlling pension benefits system / PBS PARTNERSHIP" (2000).**

> «[...] Methods only involving economic concepts and practices of doing business are not inventions within the meaning of Article 52(1) EPC. A feature of a method which concerns the use of technical means for a purely non-technical purpose and/or for processing purely non-technical information does not necessarily confer a technical character to such a method. [...] An apparatus constituting a physical entity or concrete product, suitable for performing or supporting an economic activity, is an invention within the meaning of Article 52(1) EPC. There is no basis in the EPC for distinguishing between "new features" of an invention and features of that invention which are known from the prior art when examining whether the invention concerned may be considered to be an invention within the meaning of Article 52(1) EPC. [...]»

Mit anderen Worten, reine Geschäftsverfahren sind unter dem Europäischen Patentübereinkommen nicht patentfähig. Schützbar kann jedoch eine technische Infrastruktur sein, die geeignet ist, eine wirtschaftliche Aktivität zu unterstützen.

Hier sieht es so aus, als ob sich das EPA inzwischen vom BGH hat überholen lassen. Die Entscheidung **BGH "Sprachanalyseeinrichtung" (2000)** setzt sich ausdrücklich von der Entscheidung des **EPA "Computerprogrammprodukt/IBM" (1998)** ab. Der BGH stellt fest, dass das herkömmliche Verständnis des Technikbegriffes eine industriell herstellbare und gewerblich einsetzbare Vorrichtung umfasse, zu deren Betrieb Energie eingesetzt (verbraucht) wird und innerhalb derer unterschiedliche Schaltzustände auftreten. Dies treffe auch auf einen Universalrechner zu, unbeschadet des Umstandes, dass ein Rechner in bestimmter Weise programmtechnisch eingerichtet sei. *Insgesamt besteht der Eindruck, dass beim Deutschen Patent- und Markenamt beziehungsweise beim Europäischen Patentamt Patente auf algorithmenartige Gegenstände gewährt werden, vorausgesetzt, es geht aus dem Anspruchswortlaut hervor, dass die Ausführung des Algorithmus auf einem physikalischen System erfolgt.* Nach der Auffassung des BGH genügt es, dass sich wenigstens die Zentraleinheit des Computers als physikalisches System zu erkennen gibt, wenn es den im Programm implementierten Algorithmus ausführt. Das Europäische Patentamt besteht darüber hinaus auf dem Nachweis «*weiterer (physikalischer) Effekte*» an der Rechnerperipherie.

Die in den Regularien benutzte Wendung «*Datenverarbeitungsprogramme als solche*» kann nach all dem äußerstenfalls[27] nur so verstanden werden, dass die *linguistische Erscheinungsform* des Programmcodes gemeint ist.

Die Verbindung des linguistischen Programmkonstruktes mit einer der Grammatik und Semantik der formalen Sprache des Programmcodes adäquaten Implementation einer Turing-Maschine[28] in Gestalt eines v. Neumann'schen Universalrechners[29] stellt eben nicht mehr nur ein einzelnes Programm, sondern bereits ein Datenverarbeitungssystem als Gesamtheit dar. Auch eine Gleichsetzung mit dem Begriff des Algorithmus verfehlt ihr Ziel, da der Algorithmus ebenfalls kein Datenverarbeitungsprogramm, sondern eine Abstraktion einer Klasse von unterschiedlichen Datenverarbeitungsprogrammen gleicher Funktionalität darstellt.[30]

Somit war die Einfügung des Begriffes "*Datenverarbeitungsprogramm als solches*" in das deutsche und europäische Patentrecht lediglich ein aus einem Mangel an technischem Verständnis geborener Akt *symbolischer, aber praktisch wirkungsloser Gesetzgebung.*

Falls es zur Entfernung dieser Formulierung aus dem Patentrecht kommen sollte, sind keine tiefgreifenden Veränderungen der Rechtsanwendung zu erwarten, da die Rechtsprechung es stets vermieden hat, darauf näher einzugehen.

Keinesfalls sollte in den Bestimmungen ein Bollwerk gegen die Patentfähigkeit von Software--Erfindungen gesehen werden.

2.3. Die Vereinigten Staaten von Amerika

2.3.1. Gesetzliche Grundlagen

Bereits in der **U.S.-Verfassung** findet sich ein Hinweis auf die Vollmachten des Parlamentes zum Schaffen von Patentschutz:

---

27 Mindestens ebensoviel spricht jedoch auch für die von *Hellfeld* vorgeschlagene Interpretation, der diese Formulierung als «*Extrapolation ins Nichts*» bezeichnet und weiter ausführt, jedes Computerprogramm sei seinem Wesen nach nichts anderes als die computergerechte Formulierung eines Algorithmus. Es erfülle somit einen Zweck, der außerhalb seines Seins als Programm liege. Der Begriff "Computerprogramm als solches" sei also deshalb inhaltsleer, weil es kein Computerprogramm gibt, das *nur ein solches ist und sonst nichts*. Es habe immer einen Zweck außer dem, Computerprogramm zu sein, und dieser Zweck definiere sein Wesen. **Hellfeld 1989**, S. 476.

28 Vgl. **Rechenberg/Pomberger 1997**, S. 66f.

29 Vgl. **Rechenberg/Pomberger 1997**, S. 213f.

30 Vgl. **Raden 1995**, S. 456, der den Begriff "Computerprogramme als solche" ebenfalls kritisiert. **Engel 1993** gibt einen Überblick über verschiedene Bemühungen, von einem juristisch, nicht technisch fixierten Ausgangspunkt aus den Begriff "Computerprogramm als solches" sinnvoll mit Inhalt zu füllen. Diese Versuche verlaufen sich jedoch in einem begrifflichen Irrgarten, der keine brauchbare Lösung liefert.

> **«Section 8**
>
> The Congress shall have Power [...] to promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries [...].»[31]

Die Definition patentfähiger Gegenstände findet sich in *United States Code 35, Section 101*:

> «Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.»[32]

Die Patentfähigkeit eines Erfindungsgegenstandes hängt somit zunächst davon ab, dass dieser einer der folgenden Kategorien zugeordnet werden kann:

- Verfahren
- Maschine
- Produkt
- Zusammensetzung

Interessant ist, dass – im Gegensatz zum deutschen Patentgesetz und zum EPÜ – ausdrücklich die *Nützlichkeit* des Erfindungsgegenstandes gefordert wird. Ein ausdrücklicher Negativkatalog mit einer Aufzählung dem Patentschutz nicht zugänglicher Gegenstände ist im U.S.-Patentgesetz nicht enthalten. Die Abgrenzung erfolgt durch die sich fortentwickelnde Rechtsprechung. Die Essenz daraus findet sich – den Patentschutz für software-bezogene Erfindungen betreffend – in den "Examination Guidelines for Computer−Related Inventions" des U.S.-Patent- und Markenamtes, die in Anhang A auszugsweise wiedergegeben sind.

Die Rechtsprechung der U.S.-Gerichte zur Abgrenzung patentfähiger Gegenstände besagt im Kern, dass dem Patentschutz "alles unter der Sonne, was Menschenwerk ist" zugänglich sein soll,

---

31 Im Internet: http://www.usconstitution.net/const.txt (2.12.2000).

32 Im Internet: http://www.uspto.gov/web/offices/pac/mpep/27.txt (2.12.2000).

vorausgesetzt, es fällt in eine der obengenannten vier Kategorien "gesetzeskonformer Gegenstände". Gegenstände, die nicht "nützlich" im Sinne des Patentrechtes sind, bleiben ausgeschlossen.[33]

Nach der bisherigen Praxis der Rechtsprechung sind "abstrakte Ideen", "Naturgesetze" und "Naturphänomene" nicht patentfähig. Die Prüfungsrichtlinie spricht ausdrücklich von den Problemen, die die praktische Anwendung dieser Ausschlusskriterien im Einzelfall mit sich bringt.

"Abstrakte Ideen" und "Naturgesetze" werden auch als "beschreibendes Material" bezeichnet. Im Hinblick auf Patentansprüche im Zusammenhang mit software-bezogenen Erfindungen wird eine Unterscheidung vorgenommen zwischen "*funktionalem* beschreibenden Material" und "*nicht-funktionalem* beschreibenden Material".

Im patentrechtlichen Zusammenhang besteht das "funktionale beschreibende Material" aus Datenstrukturen und Software, die einem Computer eine Funktionalität verleihen können, wenn sie auf einem computerlesbaren Datenträger gespeichert sind. Demgegenüber umfasst das "nicht-funktionale beschreibende Material" insbesondere Musik, literarische Werke und eine Zusammenschau oder bloße Anordnung von Daten.

Beide Arten von "beschreibenden Material" sind nach U.S.-Recht nicht patentfähig, wenn sie "als solche" in einem Patentanspruch stehen. Wenn "nicht-funktionales beschreibendes Material" auf einem computerlesbaren Datenträger abgespeichert ist, gilt es nicht als strukturell und funktional mit dem Datenträger zusammenhängend und wird als "nicht gesetzeskonformer Gegenstand" angesehen. Demgegenüber wird auf einem computerlesbaren Datenträger abgespeichertes "funktionales beschreibendes Material" wegen seines strukturellen und funktionalen Zusammenhanges mit dem Datenträger als "gesetzeskonformer Gegenstand" angesehen und ist somit dem Patentschutz zugänglich.

Im Ergebnis wird auf diese Weise dafür gesorgt, dass beispielsweise auf einem Datenträger wie einer CD aufgezeichnete Musik nicht patentfähig ist, denn diese fällt aus der Sicht des U.S.-Patentrechtes in die Kategorie des "nicht-funktionalen beschreibenden Materials".

2.3.2. Patente auf Geschäftsverfahren ("business methods")

Die Frage der Patentfähigkeit computer-implementierbarer algorithmischer Erfindungen ist streng zu trennen von der Beurteilung von Erfindungen mit ausschließlich solchen Merkmalen, denen aus anderen Gründen kein technischer Charakter beigemessen wird. In diesem Zusammenhang sind insbesondere die in den Vereinigten Staaten seit der **"State Street" (1998)**-Entscheidung zulässigen Merkmale, die sich auf Geschäftsverfahren beziehen, aufzuführen. In ihren Auswirkungen führt diese Entscheidung zur Erteilung von Patentansprüchen, die zumindest aus europäischer Perspektive gro-

---

33 Ein hervorragender Überblick über den Gang und die Veränderungen der amerikanischen Rechtssprechung befindet sich bei **Cohen/Lemley 2000 (2001)**, p. 8ff.

tesk erscheinen müssen.[34] Hier ein Zitat aus einem − in der Folge dieser Entscheidung − legitimen Patentanspruch:

> «**A method of a trainer teaching a cleaner, the method comprising**:
>
> (a) providing a training system for use in on the job training of worker whose job functions include the cleaning of a facility, the training system comprising a plurality of sections, each training system section containing instructions for cleaning a typical facility, each training system section limited in instructions to a specific physical job function to be performed in the facility, and each training system section containing illustrations of what is to be done and illustrations of what is not to be skipped and what is to be avoided in performing the cleaning job functions;
>
> (b) in the facility, the trainer showing the cleaner a document of a section of the training system, the document pictorially indicating what the cleaner is to do in each step of a specific job function;
>
> (c) the trainer telling the cleaner about each step pictorially depicted in the document;
>
> (d) the trainer showing the cleaner how to perform each step pictorially depicted in the document; and
>
> (e) the trainer having the cleaner perform the step in the presence of the trainer while the trainer coaches and the trainer having the cleaner tell the trainer about the step while referring to the document.»

Dieser Patentanspruch ist von jeder datenverarbeitungstechnischen Grundlage losgelöst.

---

34 **Vgl. U.S. Pat. No.** 5,851,117 **− "Building block training systems and training methods"**, December 22, 1998.

2.4. Japan

2.4.1. Gesetzliche Grundlagen und Amtspraxis

Im Gegensatz zu dem deutschen Patentgesetz, dem Europäischen Patentübereinkommen und dem U.S.-Patentgesetz fordert das japanische Patentgesetz die Technizität patentfähiger Gegenstände *expressis verbis*, indem es eine Legaldefinition des Erfindungsbegriffes aufnimmt, die den erfinderischen Gebrauch von Naturkräften impliziert.

Japanisches Patentgesetz (Auszug)[35]

«Section 2 - (Definitions)

(1) "Invention" in this Law means the highly advanced creation of technical ideas by which a law of nature is utilized.

(2) "Patented invention" in this Law means an invention for which a patent has been granted.

(3) "Working" of an invention in this Law means the following acts:

(i) in the case of an invention of a product, acts of manufacturing, using, assigning, leasing, importing or offering for assignment or lease (including displaying for the purpose of assignment or lease – hereinafter the same) of, the product;

(ii) in the case of an invention of a process, acts of using the process;

(iii) in the case of an invention of a process of manufacturing a product, acts of using, assigning, leasing, importing or offering for assignment or lease of, the product manufactured by the process, in addition to the acts mentioned in the preceding paragraph. [...]

Section 29 (Patentability of inventions)

(1) Any person who has made an invention which is industrially applicable may obtain a patent therefor, except in the case of the following inventions:

(i) inventions which were publicly known in Japan prior to the filing of the patent application;

(ii) inventions which were publicly worked in Japan prior to the filing of the patent application;

(iii) inventions which were described in a publication distributed in Japan or elsewhere prior to the filing of the patent application.

(2). Where an invention could easily have been made, prior to the filing of the patent application, by a person with ordinary skill in the art to which the invention pertains, on the basis of an invention or inventions referred to in any of the paragraphs of Subsection (1), a patent shall not be granted for such an invention notwithstanding Subsection (1). [...]

Section 32 (Unpatentable inventions)

The inventions liable to contravene public order, morality or public health shall not be patented, notwithstanding Section 29. [...]»

35 Im Internet: http://www.jpo-miti.go.jp/shoukaie/patent.htm (2.12.2000).

Aus den "Implementing Guidelines for Inventions in Specific Fields"[36] des japanischen Patentamtes, die auszugsweise in Anhang B in englischer Übersetzung des japanischen Patentamtes wiedergegeben sind, wird deutlich, dass

- eine Patenterteilung für ein "Verfahren" in Betracht kommt, wenn eine "software-bezogene Erfindung" als zeitliche Abfolge von Prozessen als zeitlich verkettete Folge von Prozessschritten ausgedrückt wird
- eine Patenterteilung für ein "Produkt" in Betracht kommt, wenn eine "software-bezogene Erfindung" in Gestalt einer oder mehrerer durch die Erfindung ausgeführter Funktionalitäten ausgedrückt werden kann

Nach japanischem Patentrecht sind auch Patentansprüche auf ein «*Speichermedium mit einem darauf aufgezeichneten Datenverarbeitungsprogramm*» grundsätzlich zulässig.

2.5. Welthandelsorganisation (WTO)

2.5.1. Allgemeines

Die Welthandelsorganisation (WTO) ist ein durch das Welthandelsabkommen begründetes eigenständiges Rechtssubjekt, das sich nicht aktiv mit der Erteilung von Patenten im Einzelfall befasst. Das Welthandelsabkommen umfasst aber einen Anhang − das "Abkommen über handelsbezogene Aspekte der Rechte des geistigen Eigentums" (TRIPS) vom 15.4.1994[37] − das allen Mitgliedsstaaten verbindliche Mindeststandards für die jeweilige nationale Legislation im Bereich des gewerblichen Rechtsschutzes vorschreibt.

2.5.2. Artikel 27 TRIPS und seine Auswirkungen

In Artikel 27 TRIPS werden alle Mitgliedsländer der Welthandelsorganisation verpflichtet, Patente für Erfindungen *auf allen Gebieten der Technik* zuzulassen, sowohl für Erzeugnisse als auch für Verfahren, vorausgesetzt, dass diese neu sind, auf einer erfinderischen Tätigkeit beruhen und gewerblich anwendbar sind. Es handelt sich um Mindeststandards, die von den Mitgliedsstaaten nicht unterboten werden dürfen. Artikel 27 TRIPS enthält keine Ausnahmebestimmungen für software-bezogene Erfindungen, die einen kategorischen Ausschluss computer-implementierbarer oder computer--implementierter Erfindungen von der Patentfähigkeit rechtfertigen könnten.

Allerdings ist der Begriff der "Technik" im TRIPS-Abkommen nicht definiert. Es könnten daher Dispute über eine weitere oder restriktivere Auslegung dieses Begriffes auftreten, falls einzelne Mitgliedsstaaten versuchen sollten, den in Kapitel 1 diskutierten Weg der "Technizitätsdiskriminierung" weiter

---

36 Im Internet: http://www.jpo-miti.go.jp/infoe/txt/soft-e.txt (2.12.2000).

37 Übersetzung aus **Hummer/Weiß 1997**.

zu verfolgen. Dabei ist zu berücksichtigen, dass jegliche Auslegung des Artikels 27 TRIPS in völkerrechtskonformer Weise *willkürfrei* und insbesondere unter Berücksichtigung der Wiener Konvention zu geschehen hätte. Eventuellen politisch motivierten Versuchen, durch eine zielgerichtete Umwertung des Technizitätsbegriffes eine wirksame Technizitätsdiskriminierung gegen die Patentfähigkeit software-bezogener Erfindungen herzustellen, sind dadurch enge Grenzen gesetzt.

Artikel 27 TRIPS

«Artikel 27 [Patentfähige Gegenstände]

(1) Vorbehaltlich der Absätze 2 und 3 ist vorzusehen, daß Patente für Erfindungen auf allen Gebieten der Technik erhältlich sind, sowohl für Erzeugnisse als auch für Verfahren, vorausgesetzt, daß sie neu sind, auf einer erfinderischen Tätigkeit beruhen und gewerblich anwendbar sind. Vorbehaltlich des Artikels 65 Absatz 4, des Artikels 70 Absatz 8 und des Absatzes 3 dieses Artikels sind Patente erhältlich und können Patentrechte ausgeübt werden, ohne daß hinsichtlich des Ortes der Erfindung, des Gebiets der Technik oder danach, ob die Erzeugnisse eingeführt oder im Land hergestellt werden, diskriminiert werden darf.

(2) Die Mitglieder können Erfindungen von der Patentierbarkeit ausschließen, wenn die Verhinderung ihrer gewerblichen Verwertung innerhalb ihres Hoheitsgebiets zum Schutz der öffentlichen Ordnung oder der guten Sitten einschließlich des Schutzes des Lebens oder der Gesundheit von Menschen, Tieren oder Pflanzen oder zur Vermeidung einer ernsten Schädigung der Umwelt notwendig ist, vorausgesetzt, daß ein solcher Ausschluß nicht nur deshalb vorgenommen wird, weil die Verwertung durch ihr Recht verboten ist.

(3) Die Mitglieder können von der Patentierbarkeit auch ausschließen

a) diagnostische, therapeutische und chirurgische Verfahren für die Behandlung von Menschen oder Tieren;

b) Pflanzen und Tiere, mit Ausnahme von Mikroorganismen, und im wesentlichen biologische Verfahren für die Züchtung von Pflanzen oder Tieren mit Ausnahme von nicht--biologischen und mikrobiologischen Verfahren. Die Mitglieder sehen jedoch den Schutz von Pflanzensorten entweder durch Patente oder durch ein wirksames System sui generis oder durch eine Kombination beider vor. Die Bestimmungen dieses Buchstabens werden vier Jahre nach dem Inkrafttreten des WTO-Übereinkommens überprüft.»

3. Der Stand der internationalen Diskussion

> «[Patent-] Legislation needs a better reason than that lawyer like it, and that America does it.»[38]

Die Diplomatische Konferenz zur Revision des Europäischen Patentübereinkommens (EPÜ), die Mitte November 2000 in München stattfand, hat zu einer Vielfalt kontroverser Stimmen im Vorfeld der Konferenz geführt. Diese Diskussionen haben – häufig unerkannt – gewissermaßen vor einem Vorhang stattgefunden, der die eigentlichen politischen und strategischen Streitfragen zwischen den großen Handelsnationen verhüllt hat:

- In europäischen Kreisen besteht eine weit verbreitete Furcht vor "amerikanischen Verhältnissen". Gemeint ist die angeblich ausufernde Praxis des amerikanischen Patentamtes, Patente auch auf "business methods" zu vergeben. Insofern reiht sich der Streit in die lange Reihe von Handelsstreitigkeiten zwischen den USA und der Europäischen Union ein.[39]
- Eine sich zaghaft bildende europäische Innenpolitik sorgt dafür, dass die Arena für Patentstreitigkeiten Brüssel ist – nicht etwa München, wo sich die eher fachlich am Patentrecht interessierten Kreise treffen. Es sind auch neue Akteure auf der Bühne, die sich auf diesem Gebiet bisher nicht zu Wort gemeldet haben.
- Die regionale Auseinandersetzung in Europa erhält ein zusätzliches Gewicht, weil ein wichtiger internationaler Akteur, die World Trade Organization (WTO), nach dem Scheitern der Konferenz von Seattle im Augenblick seine Rolle noch nicht wiedergefunden hat.

Im Vordergrund der Diskussionen stand natürlich der "Basic Proposal" zur Konferenz. Darin wurde vorgeschlagen, Art. 52 EPÜ wie folgt zu ändern:

> «European patents shall be granted for any inventions in all fields of technology, provided that they are new, involve an inventive step and are susceptible of industrial application.»[40]

Computerprogramme "als solche" werden nach diesem Vorschlag aus der Liste der nicht patentfähigen Erfindungen gestrichen (Art. 52 Abs. 2 lit. c EPÜ). Die Befürworter dieses Vorschlags verweisen

---

38 **Lessig 2000 a.**

39 Bei **Lutterbeck 2000**, S. 60, werden weitere Streitigkeiten im Bereich der Informationstechnologien behandelt.

40 Im Internet: http://www.epo.co.at/epo/dipl_conf/pdf/dm00001.pdf (1.10.2000).

dabei zumeist auf das Recht der WTO. Dieses mache eine Novellierung des EPÜ zwingend erforderlich.

Artikel 27 "Abkommen über handelsbezogene Aspekte der Rechte des geistigen Eigent (TRIPS)" vom 15.4.1994:[41]

«Artikel 27: Patentfähige Gegenstände

1. Vorbehaltlich der Absätze 2 und 3 werden Patente für alle Erfindungen, ob sie Erzeugnisse oder Verfahren betreffen, auf allen Gebieten der Technik gewährt, vorausgesetzt sie sind neu, beruhen auf einer erfinderischen Tätigkeit und sind gewerblich anwendbar.

2. Die Mitglieder können Erfindungen von der Patentierbarkeit ausschließen, wenn das Verbot ihrer gewerblichen Verwertung innerhalb eines Gebiets zum Schutz der öffentlichen Ordnung oder der guten Sitten einschließlich des Schutzes des Lebens oder der Gesundheit von Menschen, Tieren oder Pflanzen oder zur Vermeidung einer ernsten Beeinträchtigung der Umwelt notwendig ist, sofern ein solcher Ausschluß nicht nur deshalb vorgenommen wird, weil die Verwertung durch innerstaatliches Recht verboten ist.»

Allerdings enthält diese Vorschrift keine Legaldefinition des Begriffs "patentfähige Erfindung". Nach Auffassung der Wissenschaft bleibt es dann doch wieder den Mitgliedstaaten (und der Europäischen Union als Mitglied der WTO) überlassen, ob *Computerprogramme als solche* zu den patentfähigen Erfindungen gerechnet werden.[42]

Bereits unmittelbar vor der Münchner Konferenz begann sich abzuzeichnen, dass der "Basic Proposal" gegenwärtig nicht mehrheitsfähig ist. Bereits jetzt wird diskutiert, eine Nachfolgekonferenz für Ende 2002 unter dem Stichwort "Second Basket" einzuberufen.[43]

---

41 Übersetzung von **Hummer/Weiß 1997**.

42 **Senti 2000**, S. 638.

43 Im "Second Basket" soll insbesondere die Frage der Neuheitsschonfrist verhandelt werden. Bestimmte andere Kreise möchten außerdem, dass die Frage der Revision des Artikel 52 EPÜ in den "Second Basket" kommt, da eine Einigung in der Münchner Konferenz nicht erzielt werden konnte.

3.1. Diskussionen im Kräftedreieck Berlin – Brüssel – München

Politik und "interessierte Kreise"

Die wichtigsten europäischen Regulatoren waren sich schon früh über die Notwendigkeit eines Patentschutzes für Software einig: Die Kommission übernahm am 10. April 2000 die Position des Europäischen Parlaments, die es im Grünbuch über das Gemeinschaftspatent formuliert hatte:

> «Das Europäische Parlament hat sich für die Patentierbarkeit von Computerprogrammen ausgesprochen, sofern das betreffende Produkt den Anforderungen der Neuheit und der gewerblichen Anwendbarkeit genügt, wie dies auf internationaler Ebene bei unseren Wirtschaftspartnern, insbesondere den Vereinigten Staaten und Japan, der Fall ist.»[44]

In ihrem Bericht schlägt sie unter anderem folgendes Vorgehen vor:

> «Parallel zu dieser Rechtssetzungsarbeit müssen die Vertragsstaaten des Münchner Übereinkommens eine Änderung des Artikels 52 (2) Buchstabe c des Europäischen Patentübereinkommens vornehmen und Computerprogramme aus der Liste der nicht patentfähigen Erfindungen streichen. Dies ist notwendig, um die Arbeiten auf Gemeinschaftsebene und jene im Rahmen des Münchner Übereinkommens aufeinander abzustimmen.»

Andererseits war sich die Kommission bewusst, dass die zahlenmäßig wachsende "Open Source"-Bewegung politisch und fachlich möglicherweise neue Argumente auf die Tagesordnung setzen könnte. Auf Initiative der Generaldirektion Informationgesellschaft der EU-Kommission hat sich eine "EU Working Group on Free/Open Software" gegründet. Nach den ersten Stellungnahmen von Gruppen und Personen aus der "Open Source"-"Szene" musste die Kommission davon ausgehen, dass ihr Konzept in diesen Kreisen nicht zustimmungsfähig ist.[45] Besonders drängend waren Gruppen,

---

44 **Bericht zur Software-Richtlinie (91/250/EWG) 2000.**

45 Working group on Libre Software: ''Free Software / Open Source: Information Society Opportunities for Europe?'' April 2000 Version 1.2 (work in progress), im Internet: http://eu.conecta.it (3.9.2000).

die sich als ***EuroLinux*** zusammengeschlossen hatten, und ab Mitte Juni mit einer "EuroLinux--Petition" an die Öffentlichkeit getreten waren.[46]

Kurz vor der Sommerpause ließ das Europäische Patentamt seinen Vorschlag in Europa zirkulieren und konnte eine Mehrzahl der Mitglieder des EPÜ von der Richtigkeit des "Basic Proposal" überzeugen. Entsprechend hat sich der Verwaltungsrat des Europäischen Patentamtes Anfang September mit knapper Mehrheit für "Software-Patente" ausgesprochen – gegen die Stimmen Frankreichs, des Vereinigten Königreichs und der Bundesrepublik, aber mit den Stimmen kleinerer Staaten wie Zypern, Liechtenstein, Malta und der Schweiz.

Diese Vorentscheidung für "Software-Patente" im europäischen Kräftedreieck hat in der Bundesrepublik ein erstaunliches Maß an politischem Widerstand mobilisiert: Abgeordnete der Regierungskoalition haben die Entscheidung öffentlich und in Briefen an die zuständige Ministerin kritisiert,[47] der Abgeordnete Brüderle hat für die F.D.P. eine ablehnende Stellungnahme abgeben[48]. Der für den IT-Bereich zuständige Sprecher der CSU-Landesgruppe, *Martin Mayer*, ermahnte die Bundesregierung sicherzustellen, dass auf der Konferenz in München keine Ausweitung des Patentschutzes für Software beschlossen wird.[49] Seiner Meinung nach und nach Auffassung der CDU/CSU-Fraktion im Bundestag sollte die Bundesregierung ein weltweites "Moratorium" an die Stelle einer vorschnellen Entscheidung treten lassen.[50]

Man kann sagen, dass es keine relevante Stimme in der deutschen Politik gibt, die sich gegenwärtig für "Software-Patente" ausspricht. Dabei lassen die meisten Stimmen erkennen, dass die Argumente von "Open Source"-Entwicklern und -Unternehmern Eingang in die jeweiligen Positionen gefunden haben. Dies spricht für eine gute "Lobby-Arbeit" der wesentlichen Interessengruppen für "Open Source"-Software: ***EuroLinux***, ***FFII***[51] und ***FITUG***[52].

---

46 Die "EUROLINUX Alliance" versteht sich wie folgt: «The EuroLinux Alliance for a Free Information Infrastructure is an open coalition of commercial companies and non-profit associations united to promote and protect a vigorous European Software Culture based on Open Standards, Open Competition and Open Source Software such as Linux. Corporate members or sponsors of EuroLinux develop or sell software under free, semi-free and non-free licenses for operating systems such as GNU/Linux, MacOS or Windows.» Die "Petition gegen Software-Patente" findet sich im Internet unter: http://petition.eurolinux.org/index.html (2.9.2000).

47 Netzdepesche: Software-Patente − Fesseln für die Open Source? In: SPIEGEL ONLINE v. 4.11.2000, im Internet: http://www.spiegel.de/netzwelt/politik/0,1518,91895,00.html (4.11.2000).

48 Vgl. auch die Kleine Anfrage der Fraktion der F.D.P. v. 27.9.2000 "Sinn und Grenzen der Patentierbarkeit von Computersoftware" und die Antwort der Bundesregierung v. 2.11.2000 (Bundestags-Drucksache 14/4397).

49 Über die Mailingliste swpat@ffii.org (20.9.2000).

50 Antrag der Fraktion der **CDU/CSU im Deutschen Bundestag** "Sachgerechter Schutz der Rechte für Software", Bundestagsdrucksache 14/4384 v. 24.10.2000.

51 FFII: Förderverein für eine Freie Informationelle Infrastruktur (FFII e.V.), der sich selbst auffasst als einen «*gemeinnützigen Verein, in dem Projektgruppen für GNU/Linux, FreeBSD, Java, Schnittstellenspezifikationen, Normen, Lexika, Enzyklopädien, Fonts und sonstige gemeinnützige Informationswerke arbeiten können. Gemeinnützigkeit macht unsere Satzung an Merkmalen wie Schnittstellenoffenheit, Quellenoffenheit und freie Verfügbarkeit fest.*» Im Internet:

Unter den Stellungnahmen aus dem gesellschaftlichen Raum überrascht die Position der "Gesellschaft für Informatik", des größten deutschen Berufsverbandes für Informatiker mit weit über 20.000 Mitgliedern.[53] «*Zu emotional*» und «*den Sachverhalt verzerrend*» sei die aktuelle Diskussion um Pro und Contra der "Software-Patentierung". Durch den Paradigmenwechsel von «*der Kunst der Computerprogrammierung zur Softwaretechnik*», also der ingenieurmäßigen Konstruktion und Entwicklung von Software, sei dieser Zweig der Informatik zu einer Ingenieurdisziplin geworden. Die Qualität und Zuverlässigkeit heutiger, von Informatikern entwickelter Software-Systeme seien diesem Wechsel zu verdanken. Damit sei es nur folgerichtig, auch die Spielregeln der Technik auf neue Erfindungen im Bereich der Software-Entwicklung anzuwenden und für diese einen patentrechtlichen Schutz zu ermöglichen.

Einen vorläufigen Abschluss fand diese Diskussion mit klärenden Worten der ***Bundesjustizministerin***[54] und der Antwort der ***Bundesregierung*** auf eine Kleine Anfrage der F.D.P.-Fraktion im Deutschen Bundestag[55]: Die Freigabe der "Software-Patentierung" dürfe vorerst nicht umgesetzt werden.

Fachliche Stellungnahmen

Im Gegensatz zur Politik haben sich die eher fachlich orientierten Kreise in der Bundesrepublik *für* die Übernahme des "Basic Proposal" ausgesprochen.

Die **Deutsche Patentanwaltskammer** hat den Basisvorschlag begrüßt:

> «Die vorgeschlagene Änderung des Art. 52 sieht vor, den Begriff "Erfindungen" durch Hinzufügung "auf allen Gebieten der Technik" zu ergänzen und Art. 52 Abs. 2 zu streichen. Damit wird das EPÜ an Art. 27 TRIPS angepasst. Dies begrüßen wir nachdrücklich. Es entspricht einer von uns schon lange erhobenen Forderung.»[56]

---

http://www.ffii.org (20.9.2000).

52 FITUG: Förderverein für Informationstechnik und Gesellschaft. Der FITUG sieht sich wie folgt: «*Der Förderverein Informationstechnik und Gesellschaft FITUG e. V. (FITUG) schafft Verbindungen zur virtuellen Welt der Neuen Medien und der Datennetze. In unserer Satzung heißt es dazu: "Zwecke des Vereins sind die Förderung der Integration der neuen Medien in die Gesellschaft, die Aufklärung über Techniken, Risiken und Gefahren dieser Medien, sowie die Wahrung der Menschenrechte und der Verbraucherschutz in Computernetzen. Durch die genannten Zwecke sollen Kultur, Bildung und Wissenschaft gefördert werden." Der FITUG e.V. ist Mitglied im weltweiten Dachverband "Global Internet Liberty Campaign (GILC)".*» Ein Statement zur Patentpolitik findet sich im Internet unter http://www.fitug.de/news/pes/patente.html (2.11.2000).

53 Pressemitteilung v. 13.9.2000, im Internet: http://www.gi-ev.de/informatik/presse/presse_000913.shtml (18.9.2000).

54 Interview für SPIEGEL ONLINE v.27.10.2000, im Internet: http://www.spiegelonline.de/netzwelt/ (27.10.20).

55 Bundestags-Drucksache 14/4397 v. 2.11.2000.

Auch *Wolfgang Tauchert*, der zuständige Direktor beim Deutschen Patent- und Markenamt, hat sich bei verschiedenen Gelegenheiten öffentlich für die Patentierung programmbezogener Erfindungen ausgesprochen.[57] *Tauchert* argumentiert im Kern ökonomisch:

- Der Schutz des patentierten Gegenstandes als exklusives Nutzungsrecht für den Patentinhaber wird zum Teil als Belohnung für die der Gesellschaft übermittelte Information gesehen. Wirtschaftlich betrachtet werden damit die für das "Machen", d. h. für die reale Umsetzung des patentierten Gegenstandes, erforderlichen Investitionen abgesichert. Gerade Neugründungen ("Start-ups") brauchen Patente bei der Aufnahme fremden Kapitals.
- Die allgemeine Erfahrung zeigt, dass Patente eine Förderung des Wirtschaftslebens bewirken.
- Im Bereich der Software-Entwicklung entsteht mit den programmbezogen Patenten lediglich eine Situation, die mit derjenigen in anderen Gebieten der Technik vergleichbar ist. Auch dort müssen die am Markt konkurrierenden Anmelder die Patentsituation beobachten und die gesetzlich vorgesehenen Rechtsmittel zur Abwehr unberechtigter Ansprüche nutzen.

Tauchert interessiert sich für die Motive der Personen aus dem "Open Source"-Bereich, die er so charakterisiert:

> «Die Gegner des Patentschutzes verfolgen eigene Interessen, nämlich den unbeschränkten Zugriff auf das geistige Eigentum Anderer. Insbesondere die Standardformate für Daten (gif, MP3, DVD) sind dabei ins Visier genommen.»
>
> «Es hat auch in der Vergangenheit immer wieder "Pionier-Erfindungen" gegeben, die nicht zum Patent angemeldet wurden wie z.B. das Prinzip des Speicherprogramms – v. Neumann – und des Internet-Standards − Berners-Lee. Dies waren jedoch Entscheidungen von Menschen, die bewusst von sich aus auf Patentschutz verzichtet haben. Man mag sie als Wohltäter ehren und im Gedächtnis behalten. Ein allgemeiner Anspruch zum Verzicht auf eigene Rechte und zur "digitalen Enteignung" kann hieraus nicht abgeleitet werden.»

Auch *Klaus-J. Melullis*, am Bundesgerichtshof Mitglied des für Patentsachen zuständigen X. Zivilsenates und zuständig für Patentstreitigkeiten, spricht sich dafür aus, den "Basic Proposal" zu übernehmen. *Melullis* weist auf eine Selbstverständlichkeit hin, die in dem streckenweise lebhaft geführten Streit manchmal verloren gegangen ist:

---

56 Im Internet: http://www.patentanwalt.de/aktuell/revision_epue.html (1.10.2000).

57 **Tauchert 2000**.

> «[...] der Streit über diese Streichung ist daher im Ergebnis weitgehend müßig. Die bisherige, nach Wortlaut und Inhalt mißglückte weil unklare und unbestimmte Regelung schließt die Erteilung von Patenten für Programme nicht aus. Sie schafft lediglich Probleme bei der exakten Bestimmung dessen, was in diesem Bereich geschützt werden kann, insbesondere deshalb, weil sich die Diskussion auf den Begriff der Programme für Datenverarbeitungsanlagen als solcher, wie er im Abkommen gebraucht wird, fokussiert und dabei der Blick auf die sachlichen Voraussetzungen eines Patentschutzes in diesem Bereich verloren geht.»[58]

Melullis äußert eine ernste Sorge vor der Patentierung von Inhalten und erwähnt besonders eine denkbare Patentierung von Textverarbeitungsprogrammen:

> «Der Erfolg eines solchen Schutzes, wie er sich schon heute wenn auch vom Gegenstand her nicht mit einer solchen Tragweite in angemeldeten und erteilten Patenten findet, wäre nicht der technische Fortschritt, dessen Förderung das Patentrecht dient, sondern Stagnation.»

Melullis äußert sich auch zur Motivation der Entwickler von quelloffener Software. Technische Entwicklungen kosteten viel Geld. Dieser Aufwand werde natürlich nur dann getrieben, wenn das eingesetzte Kapital wieder erwirtschaftet wird.

> «Sicher hat es zu allen Zeiten Idealisten gegeben, die das Ergebnis ihrer Arbeit ohne das Verlangen nach einer Gegenleistung zur Verfügung gestellt haben. Andererseits müssen auch die Entwickler derartiger Systeme leben und benötigen dafür Geld.»

Eine vehemente Stellungnahme gegen "Open Source"-Software hat *Albert Endres* im Oktoberheft 2000 der Zeitschrift "Informatik Spektrum", dem Verbandsorgan der Gesellschaft für Informatik, abgegeben: Der Autor hat «*noch Verständnisprobleme mit dem "neuen Kult"*» und äußert tiefe Skepsis gegen die Seriosität des Anliegens:

---

58 **Melullis 2000**, S. 2.

> «In soziologischer Sicht haben Graswurzelaktivitäten natürlich enorme Vorteile, die kein Industrieprodukt so leicht wettmachen kann. Es werden dadurch u.U. sehr tief liegende Emotionen angesprochen. Wer sie ignoriert, macht sicher einen Fehler. Fast jedes technische Gebiet kann zum Betätigungsfeld von Hobbyisten werden. Wie bei allen Erzeugnissen, bei denen sich das Ergebnis professioneller und amateurhafter Vorgehensweise auf den ersten Blick kaum unterscheiden, so ist auch im Falle von Software eine Grenzüberschreitung vom Atelier zum Labor oder vom Bastler zum Ingenieur leicht möglich.»[59]

Endres fasst seine Position so zusammen:

> «So wie die Stadt Berlin nicht umhin kann, abzuwägen zwischen "Love Parade" und klassischem Kulturangebot, muss man sich auch hier überlegen, wofür man sich mehr engagiert [für industriell gefertigte Software oder für "Open Source"-Software].»[60]

3.2. Wissenschaftliche Diskussionsbeiträge

Natürlich hat es in der Vergangenheit zahlreiche Beiträge über den patentrechtlichen Schutz von Software gegeben.[61] In jüngster Zeit gibt es auch erste Beiträge, die sich mit "Open Source"-Software beschäftigen − allerdings aus der Sicht des Urheberrechts und nicht des Patentrechts.[62]

Im Bereich der Mikroökonomie ebenso wie in der Makroökonomie gibt es erste wissenschaftliche Ergebnisse, die den vorläufigen Schluss zulassen, dass es sich bei "Open Source"-Software um ein neuartiges ökonomisches Phänomen handeln könnte.

---

59 **Endres 2000**, S. 320.

60 **Endres 2000**, S. 321.

61 Vgl. etwa **Betten 1995** und **Tauchert 1999**.

62 Vgl. **Koch 2000**, **Metzger/Jäger 1999**, **Siepmann 1999**.

3.2.1. Die Sicht der Mikroökonomie

Patente als gesetzlich geschützte Marktvorteile

In einem der neuesten Hefte der Harvard Business Review beschäftigen sich *Rivette/Klein* mit Patenten als betriebswirtschaftlichen Erfolgsfaktoren.[63] Ihr Beitrag gibt auf hohem Niveau die klassische Sicht auf das Thema "Software-Patente" wieder. Der Kern ihrer Position wird deutlich aus dem englischen Original ihres Beitrags ("Discovering new value in intellectual property") – viel eher als in der reißerischen Überschrift der deutschen Übersetzung: "Wie sich aus Patenten mehr herausholen lässt". In der sicheren Erkenntnis, dass Mehrwert in einer Wissensgesellschaft durch das Managen geistiger Vermögenswerte geschaffen wird, versuchen sie zu belegen, dass man die bloß juristische Betrachtung dieser geistigen Vermögenswerte hinter sich lassen müsse. Sie fassen ihre zentrale Botschaft so zusammen:

> «Ein strategisches Managen kann den Erfolg des Unternehmens in dreifacher Hinsicht fördern:
>
> - Es wird ein patentrechtlich geschützter Marktvorteil geschaffen,
> - das finanzielle Betriebsergebnis wird verbessert und
> - die Wettbewerbsfähigkeit insgesamt gesteigert. »[64]

Sie vermuten, dass «*Patente eines der wirkungsvollsten – und manchmal sogar das wirkungsvollste − Mittel [sind], um einen rechtlich geschützten Marktvorteil zu erlangen und auch zu verteidigen*». Es komme vor allem darauf an, um die Spitzenprodukte eines Unternehmens eine «*Mauer aus Patenten*» zu ziehen. Dies bringe nicht nur unmittelbar Vermögensvorteile. Mindestens genauso bedeutsam sei es, solche Vermögenswerte in Kooperations- und Lizenzbeziehungen als Verhandlungsmasse mit anderen Unternehmen einzubringen.

In ihrem Aufsatz berichten *Rivette/Klein* von einer Spitze amerikanischer Unternehmen, die mit entsprechenden Strategien Erfolg hatten. Hierzu gehören Internet-Unternehmen wie Amazon.com und Dell, klassische Unternehmen wie IBM, Rank Xerox und Microsoft, aber auch Start-up−Unternehmen. Nach *Rivette/Klein* erwirtschaftet allein IBM aus jährlichen Patent- und Lizenzgebühren eine Milliarde Dollar:

---

63 **Rivette/Klein 2000**.

64 **Rivette/Klein 2000**, S. 29.

> «Dabei ist zu bedenken, dass es sich bei dieser Milliarde größtenteils um freien Cash-flow handelt –ein wiederkehrender Strom von Nettoeinnahmen, der für ein Neuntel des IBM-Jahresgewinns vor Steuern sorgt. Um einen vergleichbaren Nettoerlös zu erzielen, müsste IBM jährlich weitere Produkte für rund 20 Milliarden Dollar verkaufen, was einem Viertel des weltweiten Umsatzes entspräche.»

Die Autoren, Unternehmensberater und Software-Entwickler, bleiben zwar den Beweis für ihre Behauptungen schuldig, die Aussagen scheinen allerdings plausibel. Diese generelle Sicht auf Patente wird auch in deutschen Untersuchungen bestätigt. Auf einer – allerdings schmalen – empirischen Basis kommt auch der Münchner Innovationsforscher *Dietmar Harhoff* zu vergleichbaren Aussagen:

> «[...] der Patentschutz [hat] national wie international die Funktion eines strategischen Instruments angenommen, mit dem erbitterte Kämpfe um Marktpositionen ausgefochten werden.»[65]

Im Extrem hätten Patente zwei Funktionen: «*das Patent als Waffe im Wettbewerb*», sowie das Patent als Verhandlungsmasse in Lizenzierungsverhandlungen («*bargaining chip*»).

Internet-Ökonomie und Invovationswettbewerb

Einen anderen methodischen Weg beschreiten Arbeiten, die am Lehrstuhl von Prof. *Lutterbeck* in Berlin[66] und am Institut für Wirtschaftsinformatik in Saarbrücken unter der Leitung von Prof. *Scheer*[67] gefertigt wurden. Diese Arbeiten untersuchen die Entwicklungsdynamik der Herstellung und des Vertriebs von "Open Source"-Software. Der *besondere Prozess* der Herstellung und des Vertriebs von "Open Source"-Software führe zu Wettbewerbsvorteilen.

> «Da technische Neuerungen zum öffentlichen Gut werden, beschleunigt sich der Entwicklungsprozeß, wenn die Quellen frei verfügbar sind.»[68]

---

65 **Harhoff/Reitzig 1999**.

66 **Ardal 2000**.

67 **Nüttgens/Tesei 2000 a**, **Nüttgens/Tesei 2000 b**, **Nüttgens/Tesei 2000 c**.

68 **Nüttgens/Tesei 2000 c**, S.17

Dieser Prozess müsse sich zwangsläufig an den Benutzerbedürfnissen ausrichten. Wettbewerbsvorteile bestehen also hauptsächlich für die Ausnutzung des Zeitfaktors.[69] Weil Innovationen immer schneller aufeinander folgen, sehen sich Unternehmen gezwungen, immer schneller zu innovieren. Entwicklungszeiten werden damit zum Engpass. "Open Source"-Software ist in hohem Maße geeignet, solche Engpässe zu überbrücken. *Nüttgens/Tesei* verdeutlichen dieses Problem durch folgendes Beispiel:

> «Bislang war es Unternehmen möglich, eine Innovationsstrategie im Alleingang zu verfolgen, wie dies Microsoft tut. Seit einigen Jahren findet jedoch eine zunehmende Substitution uni-technologischer Innovationen durch multi-technologische Innovationen statt. Um ein neues Produkt am Markt erfolgreich zu plazieren, sind oftmals Neuerungen aus mehreren Wissensbereichen zu kombinieren. Besonders deutlich wird dies am Beispiel von E-Commerce--Projekten, die die Integration von Betriebssystemen, Webservern, Datenbanken, etc. benötigen. Diese Komplementärtechnologien sind für Unternehmen neben ihren Kernkompetenzen wichtig, um innovativ sein zu können. Um die für zukünftige Innovationen notwendigen Komplementärkompetenzen vorrätig zu halten, müßten Unternehmen breit diversifiziert sein. Dies ist jedoch mit erheblichen Koordinationskosten verbunden. Auch ist es meist nicht sinnvoll, Komplementärtechnologien kurzfristig zuzukaufen, da aktuell begehrte Technologiebasen hohe Preise haben. Für Unternehmen biete es sich daher an, bei multitechnologischen Innovationen kooperative Unternehmensnetzwerke einzugehen, die einen schnellen und kostengünstigen Zugriff auf die fehlende Komplementärtechnologie ermöglichen. Beispielsweise benötigte IBM eine Web-Server Software für E-Business-Produkte, entschied sich bewußt nicht für die Selbstentwicklung, sondern ging eine Kooperation mit der Apache-Group ein, die den benötigten Webserver bereitstellte. Der freie Quellcode des Apache-Webservers ermöglichte es IBM die Server-Software exakt auf die entsprechenden Bedürfnisse anzupassen und von Apache-Benutzern testen zu lassen.»

69 **Nüttgens/Tesei 2000 c**, S. 16/17.

Das Fazit dieser Arbeiten ist aus mikroökonomischer Sicht eindeutig:[70]

> «Das Open Source Konzept stellt eine alternative Form der Software-Entwicklung und des Softwarevertriebs dar. Dieses System kombiniert bereits vorhandenes Wissen über Softwareentwicklung, -Organisation und -Vertrieb mit den Effekten der Internet-Ökonomie. Im Vergleich zu klassischen Softwareunternehmen, üblicherweise wird hier Microsoft angeführt, wird die Entwicklung und der Vertrieb von Open Source-Software stark von den sinkenden Transaktionskosten beeinflußt. Dies führt dazu, daß in immer mehr Bereichen proprietäre Software durch Open Source-Software substituiert werden wird. Durch die zunehmende Standardisierung von lizenzkostenfreien Software-Schnittstellen und -Formaten nimmt die Bedeutung proprietärer Softwarelösungen ab. Dies kann nur durch quasi-Monopole verhindert werden. Dies bedeutet, daß Benutzer in Zukunft zunehmend weniger an große Software-Unternehmen gebunden sind, sondern die benötigte Software aus Open Source-Quellen wie z.B. anderen Unternehmen und neu entstehenden Software-Beschaffungsmärkten und -Brokern erhalten. Hierfür sind dann keine Lizenzgebühren, sondern Dienstleistungsgebühren abzuführen.»

Zwar nehmen beide Arbeiten nicht zum Problem der "Software-Patente" Stellung, doch dürfte kein Zweifel an einer entsprechenden Schlussfolgerung bestehen: Wollen (nicht nur kleine und mittlere) Unternehmen im Wettbewerb bestehen, sind sie auf die Nutzung von "Open Source"-Software angewiesen. Wer immer diesen Wettbewerb durch Monopole wie ein Patent einschränken will, muss beweisen, dass er hierfür überwiegende Gründe des Gemeinwohls auf seiner Seite weiß.

---

70 **Nüttgens/Tesei 2000c**, S.21.

3.2.2. Die Sicht der Makroökonomie

Patente als Wettbewerbsbeschränkungen, die Innovationen fördern sollen

> «Verfügungsrechte an "gefangenen Ideen" können zu demselben Ergebnis führen wie Verfügungsrechte an gefangenen Fischen bei frei zugänglicher Fischerei- (oder Erfindungs-) tätigkeit: Es werden zu viel Mittel für Erfindungen (oder für den Fischfang) eingesetzt.»[71]

Die mit diesem Satz angesprochenen Probleme des "Elends der Allmende" führen dazu, dass die Nationalökonomie vorsichtig und differenziert an die Frage herangeht, ob und unter welchen Umständen Patente einen wirtschaftlichen Grundkonflikt effizient auflösen – den «*Konflikt zwischen den gesellschaftlichen Zielen effizienter Verwendung bereits erzeugter Information und der Schaffung idealer Motivation für die Erzeugung von Information*»[72].

Völlig unabhängig davon, ob das Für und Wider von "Software-Patenten" untersucht wird, spricht deshalb sowohl aus der älteren wie auch der neueren ökonomischen Literatur eine Skepsis über den gesamtwirtschaftlichen Nutzen von Patenten.

Aus der älteren Literatur sei auf *Walter Eucken* verwiesen, einen der Wegbereiter der bundesdeutschen Marktwirtschaft, der unter der Überschrift "Wettbewerbsordnung und offene Märkte" seine Skepsis so ausdrückt:

---

71 **Richter/Furubotn 1999**, S. 131.

72 **Richter/Furubotn 1999**, S. 131.

> «Wirtschaftsformen, die mit der Wettbewerbsordnung unvereinbar sind, systemfremde Wirtschaftsformen also, entstanden oft in Anknüpfung an das moderne Patentrecht. Auch das Patentrecht gehört zu den vielen neueren Rechtsinstitutionen, die anderes verursachten, als der Gesetzgeber wollte. Seine Absicht war es, sowohl die technische Entwicklung zu fördern als auch den Erfinder zu schützen und zu belohnen.
>
> Anders als erwartet, hat das Patentrecht trotz gewisser gesetzlicher Vorsichtsmaßnahmen starke Tendenzen zur Monopolbildung und zur Konzentration in der Industrie ausgelöst. Sie ergaben sich daraus, dass das Patent ein ausschließliches Recht begründet, einen Gegenstand herzustellen, in den Verkehr zu bringen, zu gebrauchen und zu verkaufen. Zwar schließen Patente das Angebot nicht. Aber es gibt auch eine anders geartete Gruppe von Patenten, die Grundpatente, die das Angebot von Waren schließen.
>
> In doppelter Weise hat die Schließung des Angebots durch Patente konzentrationsfördernd gewirkt. Das Patent kann in einzelnen Firmen ein individuelles Monopol verleihen. Zum anderen haben Patente die Kartell- oder Konzernbildung ausgelöst und befestigt. Und diese Wirkung war wichtiger. Dabei ist nicht nur an die eigentlichen Patentkartelle gedacht. Der Austausch von Lizenzen erleichtert die Kartellbildung; die Gefahr, die ein Mitglied im Falle des Ausscheidens läuft, das Recht an gewissen Patenten zu verlieren, kittet viele Kartelle zusammen. Auch beim Aufbau der modernen Konzerne sind Patente geradezu entscheidend geworden, und zwar für ihre Ausdehnung und für den Kampf gegen Außenseiter.»[73]

Euckens Betrachtung enthält schon beinahe alle Merkmale, die die Befürworter von "Software-Patenten" für ihre Position ins Feld führen:

- **Anspornungstheorie**

  «Die Anspornungstheorie will den technischen Fortschritt durch die Aussicht auf einen entsprechenden Ertrag fördern und über den zeitweiligen Patentschutz die Ertragserwartungen des Erfinders stabilisieren.»[74]

---

73 **Eucken 1990**, S. 268f., alle Unterstreichungen von den Verfassern.

74 **Busse 1999** Rn. 58, S.16.

- **Patente als gesetzlich geschützte Marktvorteile**

  Klassisch die Auffassung des renommierten Patentrechtlers *Straus*, der im Zusammenhang mit Patenten in der Biotechnologie ausführt: «Kein Unternehmen würde in entsprechende langfristige Projekte mit hohem wirtschaftlichen Risiko investieren, wenn es das Ergebnis nicht rechtlich vor Mitbewerbern am Markt schützen könnte.»[75]

- **Patente als verkehrsfähige Wirtschaftsgüter**

  Mit Recht ist darauf hingewiesen worden, dass erst der Patentschutz Erfindungen zu voll verkehrsfähigen Gütern macht und − im Gegensatz zu Betriebsgeheimnissen auf der einen Seite und Vergütungssystemen ohne Ausschlussrecht (etwa nach der Art des früheren DDR-Wirtschaftspatents und des Erfinderscheins) auf der anderen − marktwirtschaftlichen Wirtschaftssystemen am besten entspricht.[76]

- **Patent als Waffe im Wettbewerb**

  Dies ist die klassische Position der Mikroökonomie.

Patente als Innovationshemmnisse

Die "Patent-Traditionalisten" behaupten einen Zusammenhang zwischen Patenten für Software und der Innovationsrate in den Unternehmen. In einer im Dezember 1999 vorgestellten Untersuchung haben der Ingenieur *James Bessen* und der Harvard-Ökonom *Eric Maskin* jedoch ein Modell vorgestellt, das darauf hindeutet, dass die Annahmen der "Patent-Traditionalisten" nicht zutreffen können.[77]

In Übereinstimmung mit *Bessen/Maskin* fassen wir die Argumente der Traditionalisten in einem *ökonomischen Standardmodell* zusammen:

> « The standard economic rationale for patents is to protect potential innovators from imitation and thereby give them the incentive to incur the cost of innovation. Conventional wisdom holds that, are constrained from imitating an invention, the inventor may not reap enough profit to cover that cost. Thus, even if the social benefit of invention exceeds the cost, the potential innovator without patent protection may decide against innovating *altogether*.»

Diesem Standardmodell stellen *Bessin/Maskin* ein *dynamisches* Modell gegenüber, das auf einer bestimmten Sicht der Unternehmen auf den Software-Markt beruht:

---

75 **Straus 1998**, S.29, alle Unterstreichungen von den Verfassern.

76 **Busse 1999**, Rn. 65, S.16.

77 **Bessen/Maskin 2000**.

> «For industries like software or computers, there is actually good reason to believe that imitation promotes innovation and that strong patents (long patents of broad scope) inhibit it. Society might be well served if such industries had only limited intellectual property protection. Moreover, many firms might genuinely welcome competition and the prospect of being imitated.
>
> This is because these are industries in which innovation is both sequential and complementary. By' "sequential" we mean that each successive invention builds on the preceding one in the way that Windows built on DOS. And by "complementary" we mean that each potential innovator takes a somewhat different research line and thereby enhances the overall probability that a particular goal is reached within a given time.»

Mit Hilfe dieses Modells zeigen die Autoren, dass die Ausgaben für Forschung und Entwicklung der großen amerikanischen Unternehmen gefallen sind, nachdem Gerichte die Patentfähigkeit von Software anerkannt haben. Sie zeigen mithin, dass der Patentschutz die Innovationsgeschwindidigkeit *gesenkt* hat. Aus diesen, auch durch weitere Untersuchungen belegten, Ergebnissen lässt sich folgende These für den Schutz von Software durch Patente generieren:

> Die Innovationsgeschwindigkeit auf den Software-Märkten ist umso größer, je weniger sie durch Patente und andere Monopole behindert wird.

Derart einschneidende Wettbewerbsbeschränkungen bedürfen auf jeden Fall schon aus verfassungsrechtlichen Gründen der Rechtfertigung und sollten wissenschaftlich durch handfeste empirische Daten belegt sein. Staatliche Monopole dürfen auch in der Europäischen Union nur aus Gründen des Gemeinwohls eingeführt oder aufrecht erhalten werden. Es überrascht daher, dass die Argumente der "Patent-Traditionalisten" geradezu "Mantra-artig" in der Literatur und in einschlägigen Gerichtsentscheidungen wiederholt werden – in der juristischen Literatur häufiger als in der ökonomischen. Einen harten Beweis dafür, dass die Position der Befürworter zutreffen könnte, haben wir in der gesamten deutschsprachigen und englischsprachigen Literatur nicht finden können. Einen solchen Beweis erbringt auch die von der Europäischen Kommission am 19.10.2000 veröffentlichten Studie von *Hart/Holmes/Reid* nicht. Auch sie verschieben unzulässig die Beweislast:

> «There is no evidence that European independent software developers have been unduly affected by the patent positions of large companies or indeed of other software developers.»[78]

Auch übernehmen sie wie selbstverständlich die Belohnungstheorie. Den Beweis dafür bleiben sie ebenfalls schuldig.

Da es keine zwingenden Beweise für die Stichhaltigkeit der Argumente der "Patent-Traditionalisten" gibt, andererseits die Forschungen von *Bessen/Maskin* starke Argumente für die Richtigkeit der Gegenthese nahelegen, muss es aus Rechtsgründen zu einer Umkehrung der Beweislast kommen. Dies ist auch die Position von *Lawrence Lessig*, dem führenden Cyberlaw-Rechtswissenschaftler der USA. Er bezeichnet das dynamische Modell von *Bessen/Maskin* als «*powerful*»[79] und bewertet diese Forschungen wie folgt:

> «If there were good reason to believe that patents would improve innovation in software, the arguments of the open code movement might be weak. But no reason has yet been offered. The push for patents comes from lawyers, not technologists or economists. And the only sure effect of the change will be to increase the work for lawyers, as they increase the burden on technologists.
>
> If patenting software will induce more innovation in software development, then let proponents demonstrate it, through careful and convincing economic evidence. But until they do, Europe should wait. Legislation needs a better reason than that lawyers like it, and that America does it.»

Für *Lessig* sind die gesetzgeberischen Anstrengungen der europäischen Gesetzgeber nichts anderes als «*me-too-patent-law*».

---

78 **Hart/Holmes/Reid 2000**.

79 **Lessig 2000 a**, alle Unterstreichungen von den Verfassern. In **Lessig 2000 b** wird er noch deutlicher: «*Thus [is] the nature of the patent cycle: a process created by lawyers that benefits only one group with any real certainty - lawyers.*»

Es ist nicht ersichtlich, warum *Lessig´*s Argumente nicht auf die Bundesrepublik Deutschland übertragbar sein sollten.

3.3. Zusammenfassung

Die Diskussion um "Software-Patente" ist defizitär.

Von den Fachleuten der "Beteiligten Kreise" wird sie häufig technizitisch geführt. Meist wird übersehen, dass sich die Konsequenzen von Schutzrechtsregelungen nur auf der makroökonomischen Ebene beurteilen lassen. Die "Patent-Traditionalisten" belassen es zumeist bei dem pauschalen Hinweis auf die Anspornungstheorie. Den Beweis, dass diese Annahmen auch auf den Software-Markt zutreffen, bleiben sie schuldig.

Es gibt andererseits starke Belege für die Annahme, dass die Innovationsrate auf den Software-Märkten in dem Maße steigt, wie (starke) Patente *gerade nicht* vergeben werden. Die These "Patente behindern den technischen Fortschritt auf den Software-Märkten" ist mithin sehr viel wahrscheinlicher zutreffend als ihr Gegenteil.

Dies würde mit den Ergebnissen anderer Untersuchungen übereinstimmen, wonach die Innovationsgeschwindigkeit und damit die Wettbewerbsfähigkeit von Unternehmen in dem Maße steigen kann, wie sie quelloffene Software verwenden. Diese Auffassung öffnet zugleich eine andere und neue Sicht auf "Open Source"-Software und deren Entwickler: Es geht nicht um irgendwelche «*Graswurzelbewegungen*» nicht um Spinner, nicht um selbstgenügsame junge Leute, auch nicht um verkappte Revolutionäre, die die Eigentumsordnung umstoßen wollen.

Die fehlerhafte Einordnung ist sicher auch das Verdienst mancher Akteure in der "Open Source"-Bewegung. Nicht zuletzt ist es Folge einer Schwierigkeit der englischen Sprache. Das englische Wort "free" bedeutet sowohl *frei* und *offen* als auch *umsonst* und *gratis*. Irrtümlich nehmen deshalb manche an, dass freie Software kostenlos sein müsse. Das führt dann manche Persönlichkeiten zu der Auffassung, dass die Verfechter freier Software diese als öffentliches Gut betrachten und den Unternehmen keinen Gewinn zugestehen.

Das Gegenteil ist der Fall.

Der Finne *Linus Torvalds*, der zentrale Entwickler des LINUX-Betriebssystems, hat das vor einigen Wochen in einem Interview mit der Computerwoche klargestellt. Er war gefragt worden: «*Was halten Sie von der Debatte um Copyrights und Patente?*» Seine Antwort war differenziert:

> «Ich halte Copyrights auf Software für eine gute Sache, aber nicht Patente. [...] In der Öffentlichkeit ist der falsche Eindruck erweckt worden, die Gegner von Patenten seien gegen geistiges Eigentum. Das Hauptproblem ist, dass unsere Patentsysteme heute in der Praxis nicht funktionieren. Patente funktionieren bei ingenieurmäßiger Entwicklung, aber nicht in der Wissenschaft. Software-Patente sind genauso lächerlich wie Patente auf mathematische Formeln.»[80]

Es geht um ein alternatives Modell der Software-Entwicklung und des Software-Vertriebs. Dieses Modell ist in hohem Maße den Bedingungen der Internet-Ökonomie angepasst. Unternehmen − nicht etwa nur kleine und mittlere − riskieren Wettbewerbsnachteile, wenn sie "Open Source"-Software nicht anwenden.

Dies bedeutet, dass viele Persönlichkeiten ihre ablehnende Haltung werden überdenken müssen. Es ist eine neue Entwicklung zu verstehen, die *Nüttgens/Tesei* in dem letzten Satz ihrer Untersuchungen so beschreiben:

> «Open Source vereint gesellschaftliche, marktwirtschaftliche und technische Entwicklungen des ausgehenden 20. Jahrhunderts und bietet sich daher als neues Modell der Softwareentwicklung und des Softwarevertriebs an. Diese Entwicklung steht erst am Anfang.»[81]

Es daher auch erst in Umrissen klar, wie sich diese Entwicklung wissenschaftlich beschreiben läßt. In Kapitel 4 geben wir dazu erste Hinweise.

Zum Thema des Zusammenhanges von IT-Sicherheit und Patenten erbringt die Sichtung der nationalen und internationalen Literatur überhaupt keinen Ertrag. Wir gehen jedoch davon aus, dass es einen deutlichen Zusammenhang gibt und behandeln diese Frage in Kapitel 6.

---

80 Interview mit der Computerwoche v. 13.10.2000, im Internet: http://www.computerwoche.de/online-specials/artikel/main.cfm?id=22966 (1.11.2000).

81 **Nüttgens/Tesei 2000 c**, S. 22.

4. Proprietäre Software und "Open Source"−Software im Vergleich

Das folgende Kapitel betrachtet proprietäre Software und "Open Source"-Software vorrangig unter informatisch-technischen und ökonomischen Aspekten. Andere Randbedingungen, wie Fragen der Organisation und rechtliche Probleme, werden nur kurz angerissen, soweit es für das Verständnis notwendig erscheint. Um auch dem nicht informatisch vorgebildeten Leser das Verständnis der technischen Ausführungen zu ermöglichen, war es an den entsprechenden Stellen unvermeidlich, kurze Beschreibungen der Tätigkeiten eines Informatikers bzw. Software-Entwicklers zu integrieren.

Praktisch ausgespart bleiben historische Hintergründe, die zwar für das Verständnis des Themas wertvoll, allerdings auch besser untersucht sind als andere Aspekte. Wir verweisen den eher historisch interessierten Leser auf die Arbeiten **Raymond 1999**, **Stallman 1999**, **Gehring 1996** und die informativen Aufsätze in **DiBona/Ockman/Stone 1999**.

Für den Zweck dieses Gutachtens verzichten wir auf eine detaillierte Erarbeitung der Begrifflichkeit von "proprietärer Software" und "Open Source"-Software. Wir gehen von folgendem Verständnis aus:

"Proprietäre Software" ist ein Sammelbegriff. Es gibt keine allgemein anerkannte Definition dafür, was unter "proprietärer Software" zu verstehen ist. Seine Bedeutung ergibt sich erst aus einem Geschäftsmodell im Kontext des Urheberrechts bzw. Copyrights. Die populäre Erklärung des Begriffes lautet so:

> «By definition, proprietary software means that it isn't yours to give − someone else makes their living by selling it.»[82]

Exakter wird das Wesen proprietärer Software folgendermaßen beschrieben:

82 Vgl. **NCES 1998**, chapter 7.

Proprietäre Software ist Software, die durch den Urheberrechts- bzw. Copyright-Schutz Eigentumscharakter erlangt und so als Ware auf dem Software-Markt marktfähig wird. Verkauft werden Nutzungslizenzen für Vervielfältigungsstücke der Software. Der rechtmäßige Nutzer, d.h. der Lizenznehmer von proprietärer Software, hat prinzipiell nicht das Recht, diese zu seinen Zwecken zu bearbeiten oder Kopien zu erstellen und zu vertreiben. Durch die Lizenz wird sichergestellt, dass der Quelltext von proprietärer Software der Allgemeinheit nicht zur Verfügung steht.

Auch **"Open Source"-Software** ist ein Sammelbegriff. Im weiteren Sinne wird er heutzutage häufig auch auf Software angewandt, die von ihren Entwicklern selbst als "Freie Software" bezeichnet wird. In der engeren Bedeutung umfasst er Software, deren Nutzungslizenz den Anforderungen der "Open Source Definition" entspricht.

Open Source Definition[83]

«The Open Source Definition

(Version 1.7)

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost -- preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

83 **OSD 2000**.

4. Integrity of The Author's Source Code.

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups.

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor.

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License.

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product.

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Contaminate Other Software.

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.»

Die "Open Source Definition" ist eine Art Muster für die Erarbeitung von "Open Source"-Lizenzen. Jede Lizenz, die den Richtlinien aus der "Open Source Definition" entspricht, darf sich "Open Source"-Lizenz nennen und kann als solche von der "Open Source Initiative"[84] (OSI) zertifiziert werden. Eine zertifizierte "Open Source"-Lizenz darf die Bezeichnung "OSI Certified" tragen.[85] Die Nutzer von Software mit derartig zertifizierter Lizenz können somit sicher sein, über die in der "Open Source"-Definition genannten Nutzungsrechte zu verfügen. Die wohl prominenteste Lizenz, die den Kriterien der OSD genügt, ist die "General Public License" (GPL) der Free Software Foundation,[86] wenn es sich dabei auch um *die* "Free Software"-Lizenz handelt.[87]

---

84 Die Open Source Initatiative im Internet: http://www.osi.org (9.11.2000).

85 Diese Bedingung könnte zu einem Zeitpunkt, da "Open Source"-Software größere Bedeutung im Software-Markt erlangt hat, als es heute der Fall ist, aus kartellrechtlicher Perspektive als Konditionenkartell problematisch werden. So jedenfalls **Koch 2000b**, S. 342.

86 Die GPL wurde vom Programmierer *Richard Stallman* für das GNU-Projekt der Free Software Foundation (im Internet: http://www.fsf.org), d.h. als Lizenz für "Free Software", nicht für "Open Source"-Software, geschaffen. Die Funktion der GPL beschreiben die Autoren des "Internet Junkbusters" (http://junkbusters.com) so: «By making your DECLARATION available to them under the GPL, you are permitting them use to it, but never to claim it as their property, even if they transform it.»

87 Etwas verallgemeinernd kann gesagt werden, dass jede "Free Software"-Lizenz dem Lizenznehmer *mindestens* soviel Rechte gewährt, wie jede "Open Source"-Software-Lizenz.

Es lässt sich folgende Arbeitsdefinition für "Open Source"-Software geben:

> **"Open Source"-Software ist Software, die durch Urheberrechts- bzw. Copyright geschützt ist, ohne Eigentumscharakter zu erlangen. Der rechtmäßige Nutzer, d.h. der Lizenznehmer von "Open Source"-Software, hat das Recht, diese zu bearbeiten, bearbeitete oder unbearbeitete Kopien zu erstellen und zu vertreiben. Der Lizenznehmer hat nicht das Recht, bearbeitete oder unbearbeitete Kopien ohne Zurverfügungstellung des Quelltextes zu verteilen. Durch die Lizenz wird sichergestellt, dass der Quelltext von "Open Source"-Software der Allgemeinheit jederzeit zur Verfügung steht.**

Das Lizenzmodell von "Open Source"-Software ist ohne das Urheberrecht bzw. Copyright nicht denkbar. Anders lautende Vermutungen sind unzutreffend.

In der Praxis gilt im Übrigen: Unter mikroökonomischen Gesichtspunkten sind "Freie Software" und "Open Source"-Software Synonyme.

4.1. Aktuelle Marktdaten zu "Open Source"−Software

Dieser Abschnitt stellt einige der verfügbaren Informationen über die wirtschaftliche Bedeutung von "Open Source"-Software vor. Da es bisher keine wirklich umfangreichen Untersuchungen der Thematik gegeben hat und die Dynamik der "Open Source"-Integration in den Wirtschaftskreislauf sehr groß ist, müssen die Daten mit der entsprechenden Vorsicht bewertet werden. Aufgrund der herrschenden Begriffsunklarheiten wird in den Studien vielfach keine Unterscheidung zwischen "freier" Software und "Open Source"-Software getroffen. Die Termini werden deshalb im Folgenden bevorzugt so verwendet wie in den jeweiligen Studien.

(A) Betriebssysteme

Eine der aktuellsten Untersuchungen wurde als Leserbefragung der deutschen Fachzeitschrift iX durchgeführt.[88] Zur Auswertung konnten 1050 Antworten herangezogen werden. Da sich die

88 Vgl. **Seeger 2000**.

Zeitschrift vorrangig an ein professionelles Publikum wendet, dürften die Ergebnisse durchaus repräsentativ sein.

In 74% der Unternehmen der Leser wird das "Open-Source"-Betriebssystem Linux eingesetzt. Dabei kommt Linux in 5% der Fälle als Desktop-Betriebssystem zum Einsatz, überwiegend jedoch als Server. Linux hat seinen Verbreitungsgrad damit um 16 Prozentpunkte gegenüber 1998 und um 29 Prozentpunkte gegenüber der iX-Befragung von 1996 gesteigert.

Die meistverwendete Linux-Distribution[89] ist die SuSE-Distribution der deutschen SuSE GmbH[90]. Sie kommt auf einen Marktanteil von 72%. Die Distribution des amerikanischen Anbieters Red Hat kommt auf 34%. Andere Distributionen (Debian, Slackware etc.) kommen auf zusammen 16% Marktanteil. Die Summe von über 100% ist dadurch zu erklären, dass in vielen Unternehmen mehrere Distributionen eingesetzt werden.

Andere "Open Source"-Betriebssysteme spielen mit einer Verbreitung unter 10% insgesamt nur eine untergeordnete Rolle .[91]

Von den kommerziellen UNIX-Derivaten ist lediglich für Solaris von SUN Microsystems, AIX von IBM und für HP-UX von Hewlett-Packard eine nennenswerte, wenn auch − gegenüber Linux − deutlich geringere Verbreitung zu verzeichnen: Solaris hat 38%, HP-UX 17% und AIX 16% Verbreitung. Lediglich der Anteil von Solaris ist seit 1998 nicht gefallen.

Betrachtet man statt aller Unternehmen nur die Firmen mit mehr als 1.000 Mitarbeitern, so erreichen die kommerziellen UNIX-Derivate höhere Verbreitungsanteile. Aber auch hier führt Linux mit 58% das Feld an und weist die höchsten Zuwachsraten auf.

Die weitaus höchste Verbreitung haben Betriebssysteme aus dem Hause Microsoft aufzuweisen. Sie werden in 88% aller Unternehmen bzw. in mehr als 90% der Unternehmen mit mehr als 1.000 Mitarbeitern eingesetzt. Tendenziell neigen also kleine und mittlere Unternehmen stärker zum Einsatz von "Open Source"-Software als große Unternehmen.

Eine Untersuchung der Firma iKu Netzwerklösungen zeigt, dass im Juni 2000 ca. 44% der in Deutschland am Internet angeschlossenen Computer mit Linux als Betriebssystem arbeiteten, ca. 30% arbeiteten mit einem Microsoft-Betriebssystem.[92] Betrachtet man nicht alle Computer, sondern nur solche, auf denen eine Domain gehostet (verwaltet) wird,[93] so steigt der Marktanteil von Linux

---

89 Im deutschen Sprachraum, im dem iX verbreitet wird; Anm. der Verfasser.

90 Die SuSE-GmbH im Internet: http://www.suse.de (18.10.2000).

91 In bestimmten Bereichen, zum Beispiel bei Internet-Service-Providern, kommen bekanntermaßen bevorzugt "Open Source"-BSD-Derivate zum Einsatz. Das wurde in den Studien aber nicht separat begutachtet.

92 iKu-Netzanalyse, im Internet: http://www.iku-netz.de/netscan/ (07.11.2000).

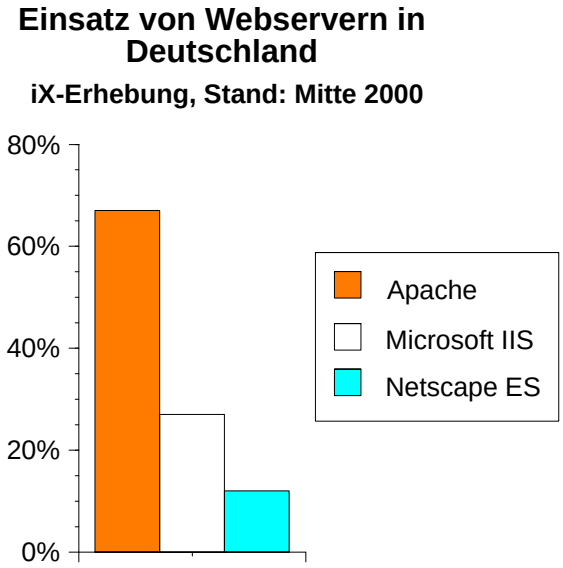
93 Ein Server kann mehr als eine Domain hosten; z.B. könnten die Domains **`domain1.de`** und

auf 48%, während Microsoft-Systeme nicht einmal mehr auf 10% kommen. Im Server-Bereich wird demnach stärker auf Linux gesetzt als im Desktop-Bereich.

Die etwas ältere Auszählung des Internet Operating System Counter (April 1999) wies demgegenüber für Linux auf Computern mit Internetanschluss eine Verbreitung von 43% in Deutschland aus.[94]

(B) Webserver

Die iX-Erhebung weist für den "Open Source"-Webserver Apache einen Marktanteil von 67% aller Webserver aus. Der proprietäre Microsoft Internet Information Server (IIS) erreicht 27%. Der ebenfalls proprietäre Netscape Enterprise-Server (ES) wird in 12% der Fälle eingesetzt. Angaben über andere Webserver fehlen.



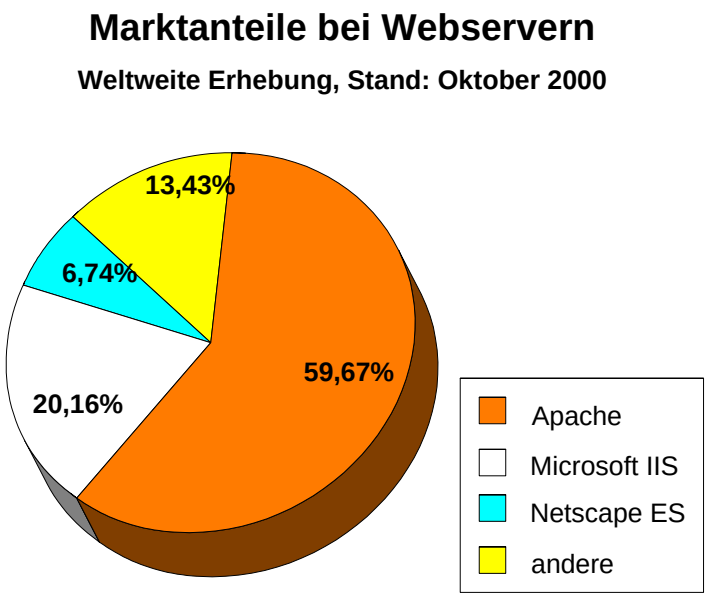
Die Summe ergibt mehr als 100% wahrscheinlich, weil etliche Firmen mehrere unterschiedliche Webserver im Einsatz haben.

---

domain2.de auf ein und demselben Server verwaltet werden.

94 Internet Operating System Counter, April 2000. Im Internet: http://www.leb.net/hzo/ioscount/data/top.domains.9904.linux.txt (07.11.2000).

Andere Untersuchungen bestätigen in etwa diese Ergebnisse. Die regelmäßig von der Firma Netcraft durchgeführte Untersuchung zur Verbreitung spezifischer Webserver zeigt für den Oktober 2000 folgende Marktanteile:[95]



Im Detail sehen die Zahlen so aus:

Webserver	Installationen	Anteil in Prozent
Apache	13.295.499	59,67%
Microsoft IIS	4.491.609	20,16%
Netscape ES	1.500.988	6,74%

Alle anderen Webserver haben weniger als 5% Marktanteil.

Ausweislich der iX-Befragung hat der "Open Source"-Webserver Apache im deutschsprachigen Raum eine höhere Verbreitung als im Weltmaßstab. Dieses Ergebnis deckt sich mit den Ergebnissen anderer Erhebungen über den Umfang des Einsatzes von "Open Source"-Software in

95 The Netcraft Web Server Survey, October 2000. Im Internet: http://www.netcraft.com/survey/ (07.11.2000).

Deutschland.[96]

(C) "Open Source"-Software allgemein

Die bereits erwähnte iX-Umfrage weist aus, dass ca. 70% der Unternehmen bereits freie Software einsetzen, weitere 6% planen den Einsatz. Zum Vergleich: In der 98er iX-Umfrage meldeten nur 58% der Unternehmen den Einsatz von freier Software.

Eine Studie der Unternehmensberatung Forrester Research[97], in der 2.500 Unternehmen befragt wurden, zeigt eine deutliche geringere Verbreitung freier Software auf: 56% der Unternehmen setzen bereits "Open Source"-Software ein, 6% planen den Einsatz. Da sich die Forrester-Studie primär auf den US-Markt und multinationale Firmen bezieht, bestätigt sie unmittelbar die relativ größere Verbreitung von freier und "Open Source"-Software in Deutschland.

(**D**) **Arbeitsmarkt**

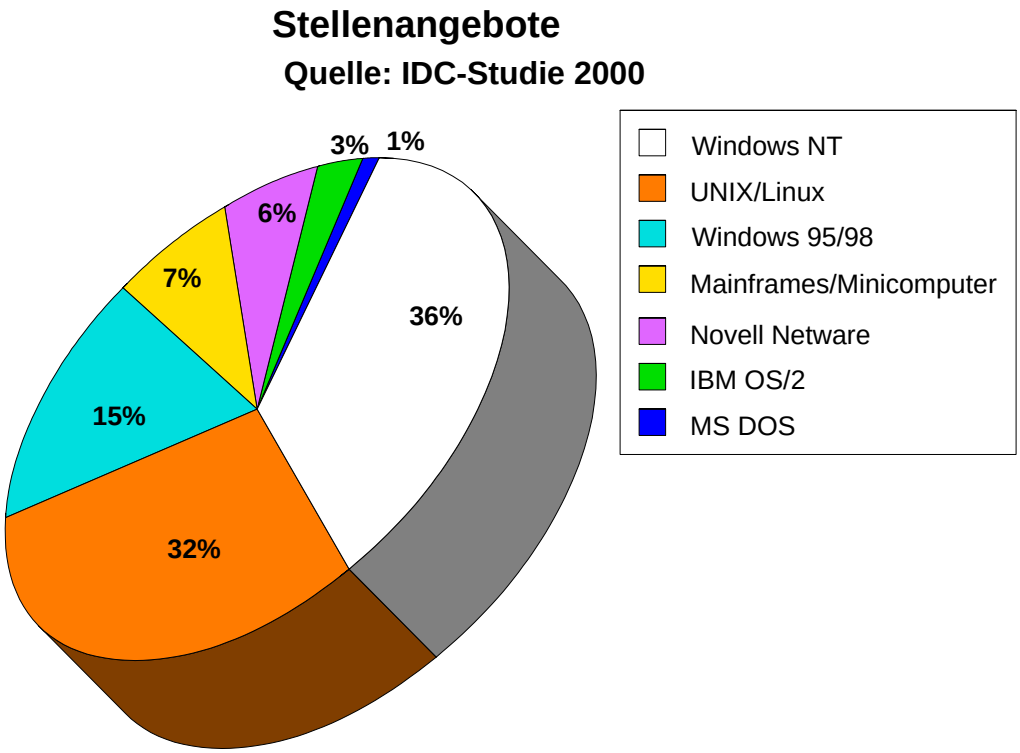
Der breite Einsatz von "Open Source"-Software bleibt nicht ohne Auswirkungen auf den Arbeitsmarkt. Die jüngste CDI-Stellenmarktanalyse,[98] die 8.650 Stellenangebote (Druck- und Online-Anzeigen) nach den darin gefragten Qualifikationen untersucht hat, weist im Bereich Betriebssysteme eine 36%ige Nachfrage nach Windows NT-Kenntnissen und eine 32%ige Nachfrage nach UNIX/Linux-Kenntnissen aus. In 15% der Fälle wurden Windows 95/98-Kenntnisse, in 7% Kenntnisse von Mainframes und Minicomputer und in 6% Novell NetWare-Kenntnisse verlangt. Andere Betriebssystemkenntnisse wurden jeweils in weniger als 5% der Anzeigen erwähnt.

---

96 Vgl. **Dempsey et al. 1999**.

97 Vgl. **Delio 2000**.

98 Vgl. **Born 2000**.



(E) Fazit

Es bleibt festzustellen, dass es an statistischen Untersuchungen zum Einsatz von "Open Source"-Software mangelt. Das vorhandene Zahlenmaterial ist eher dürftig und es besteht weiterer Bedarf an aktuellem, aussagekräftigem Datenmaterial.

Unter Beachtung des Datenmangels weisen die vorhandenen Informationen auf einen sehr hohen Einsatz von freier und "Open Source"-Software besonders in deutschen Unternehmen hin. Große Unternehmen in Deutschland und den USA unterscheiden sich in Bezug auf den Einsatz von freier Software und "Open Source"-Software nicht. Tendenziell wird freie und "Open Source"-Software stärker in kleinen und mittleren Unternehmen eingesetzt. Ob das auf die Verfügbarkeit des Quellcodes, die fehlenden Lizenzkosten oder auf andere Faktoren zurückzuführen ist, lässt sich anhand des vorhanden Materials nicht eruieren. Hier besteht Bedarf für weitergehende Untersuchungen.

Die Wachstumsraten beim Einsatz von freier und "Open Source"-Software sind beachtlich hoch und liegen besonders in den Bereichen der Internet-Ökonomie über dem Durchschnitt. Die Software-Ausstattung von Unternehmen, die im Sektor der New Economy tätig sind, besteht zum überwiegenden Teil[99] aus freier Software und "Open Source"-Software.

---

99 Bemessen nach Einsatz von "Code", nicht nach Kosten.

4.2. Die "Open Source"−Bewegung

> «Beispielsweise haben, worauf auch die Vertreter der neuen Wachstumstheorie besonders deutlich hinweisen, viele neue Erkenntnisse und Wissensbestände grundlegender Art häufig den Charakter eines öffentlichen Gutes oder sind mit starken positiven externen Effekten verbunden, weil von ihrer Nutzung niemand ausgeschlossen werden kann, oder es gibt zumindest sehr erhebliche positive externe Effekte, weil auch andere, die über diese Erkenntnisse und Wissensbestände selbst nicht unmittelbar verfügen, Vorteile davon haben, sodass der soziale Nutzen über den Nutzen, den sich die Träger des Humankapitals anzueignen vermögen, hinausgeht.»[100]

Das vorangehende Zitat stammt nicht etwa aus einem Aufsatz eines "Open Source"-Autors; es stammt aus einem klassischen Lehrbuch der Ökonomie. Die Beschreibung von *Behrens* passt jedoch auch auf das, was heute "Open Source"-Bewegung genannt wird.

Man darf bei "Open Source" nicht nur an Software denken. Die "Open Source"-Bewegung umfasst weit mehr als ein neues Modell der Software-Entwicklung: Die "Open Source"-Bewegung entwickelt Software[101], Standards[102], Content[103], Geschäftsmodelle[104], Wissen und Ideen[105]. Bereits die Selbstbeschreibung als "Community", wie sie in praktisch allen Aufsätzen von Akteuren selbst vorgenommen wird, weist auf die sozialen Aspekte hin.

Die am Institut für Wirtschaftsinformatik in Saarbrücken arbeitenden Ökonomen *Markus Nüttgens* und *Enrico Tesei* haben eine umfangreiche Untersuchung über Marktmodelle und das Community-Konzept von "Open Source"-Software vorgelegt. Sie fassen die Charakteristika der Communities,

---

100 **Behrens 2000**, S. 433

101 Der Beispiele sind unzählige: Linux, TeX, OpenBSD, GIMP... Wer sich einen kleinen Überblick verschaffen möchte, sei auf http://freshmeat.net (20.10.2000) verwiesen.

102 Die Internet-Standards (RFC's) entstehen als offene Standards in einem Prozess, der analog zum "Open Source"-Software-Entwicklungsprozess verläuft. Zuerst werden Vorschläge zur Standardisierung als "Request for Comment" veröffentlicht. Anschließend werden sie von der Internet-Öffentlichkeit diskutiert, überarbeitet und beschlossen.

103 Beispielsweise sei hier das deutsche "Projekt Gutenberg" erwähnt, im Internet: http://www.gutenberg.de (19.10.2000).

104 Vgl. u.a.: **Young/Goldman Rohm 2000**. Siehe auch die Ansätze zum Projektmanagement bei http://sourceforge.net (3.12.2000) oder http://www.collab.net (20.10.2000).

105 Vgl. u.a.: *Eric S. Raymond*: Homesteading the Noosphere. In: **Raymond 1999**, S.81-135. Im Internet: http://www.tuxedo.org/~esr/writings/homesteading (18.10.2000).

aus denen sich die "Open Source"-Bewegung formt, in folgender Tabelle zusammen:[106]

Gemeinsamkeiten	Unterschiede
• Internationale Zusammensetzung der Mitglieder	• Phase der Entstehung
• Räumlich, meist global verteilte Mitglieder	• Aktuelle Anzahl der aktiven Mitglieder
• Internet als Kommunikationskanal	• Art und Anzahl der Benutzer
• Freiwilliges und unbezahltes Engagement der meisten Mitglieder	• Art und Anzahl der unterstützten Plattformen
• Sehr wenig bezahlte Entwickler	• Lizenzpolitik
• In bestimmten Phasen extrem hohe Entwicklungsgeschwindigkeit und -dynamik	• Kooperationspolitik mit kommerziellen Interessengruppen
• Hohe Ansprüche an Qualität, Flexibilität und Stabilität der Software	
• Finanzielle Unterstützung durch Spenden	

Zu ähnlichen Kriterien und Ergebnissen kommt auch der Ökonom *Atila Ardal*, der seine Untersuchungen im Rahmen der Forschungsgruppe "Internet Governance" bei *Bernd Lutterbeck* dargelegt hat. *Ardal* hat die Unterschiede und Gemeinsamkeiten in umfangreichen Tabellen zusammengestellt.[107]

Wie bereits betont, geht es bei "Open Source" um wesentlich mehr als nur um Software oder andere "immaterielle Ware". Auch "Open Source"-Hardware ist inzwischen in Reichweite der "Open Source"-Bewegung. Aufgrund der mittlerweile auch in der Hardware-Entwicklung etablierten modulbasierten Arbeitsweise ist es verhältnismäßig einfach möglich, die im "Open Source"-Software-Entwicklungsprozess erprobten Entwicklungsmethoden auf Hardware zu erweitern und somit "Open Source"-Hardware[108]zu entwickeln.

---

106 Vgl. **Nüttgens/Tesei 2000 a**, S. 11.

107 Vgl. **Ardal 2000**.

108 **Schmid 2000** gibt einen guten Einblick in die Motivation und Verbreitung der "Open Source"-Hardware"-Entwicklung. Namhafte Unternehmen wie Gray Research und SUN Microsystems, aber auch die Europäische Raumfahrtagentur ESA, initiierten "Open Source"-Hardware-Projete. Eines der ambitioniertesten "Open Source"-Hardware-Projekte ist sicherlich das FreeCPU-Projekt, das es sich zum Ziel gesetzt hat, einen 64-Bit-Prozessor zu entwickeln, dessen Bestandteile (Architektur, Befehlssatz, Schaltungslayout) als "Open Source"-Hardware frei im Internet verfügbar gemacht werden: http://www.f-cpu.org (24.10.2000). Dass solche Arbeiten durchaus erfolgreich verlaufen, zeigt die bevorstehende Produktion des OpenRISC-1000-

Die tragende Rolle von Software innerhalb der "Open Source"-Bewegung widerspiegelt die Tatsache, dass Software aus dem modernen Kommunikationsgeschehen der "Informationsgesellschaft" nicht mehr wegzudenken ist. Die Software, die jemand verwendet, entscheidet wesentlich über dessen Möglichkeiten zur Kommunikation und die Art der Teilnahme an Communities.[109] Entwicklung von "Open Source"-Software kann in diesem Sinne als eine notwendige **Infrastrukturmaßnahme** betrachtet werden: "Open Source"-Software implementiert offene Kommunikationsstandards, darauf basierende Programme und so die Voraussetzungen für andere Aktivitäten im Rahmen der "Open Source"-Bewegung.[110]

Es sei darauf hingewiesen, dass das "Open Source"-Software-Entwicklungsmodell weitaus älter ist als vielfach angenommen. Bereits die Software-technischen Grundlagen für das Internet wurden zu großen Teilen in einem vergleichbaren Prozess entwickelt.[111] Die daraus hervorgegangene Software − z.B. BIND als Implementation des Domain Name System (DNS) für die Namensverwaltung oder das Simple Mail Transfer Protocol (SMTP) mit entsprechenden Implementationen (z.B. sendmail) für den E-Mail-Versand − existiert seit der Frühzeit des Internet und bildet noch heute dessen kommunikationstechnisches Rückgrat.

Der Kern aller dieser anderen Aktivitäten kann mit einem Stichwort beschrieben werden: **Wissenstransfer**. Die "Open Source"-Bewegung orientiert sich in all ihren Facetten am Grundsatz des **ungehinderten Wissenstransfers**.

Der ungehinderter Wissenstransfer ist das Paradigma der "Open Source"-Bewegung.

Die Lizenzbedingen für "Open Source"-Software (freie Software) formulieren diese Aussage bezogen auf Software in der Sprache der Juristen:

Prozessors der OpenCores-Initiative: http://www.opencores.org (24.10.2000).

109 Vgl. **Lessig 1999**, besonders Kap. 4 und 5.

110 Siehe z.B. verschiedene Arbeiten des Stanford-Juristen *Lawrence Lessig* zum Thema, verfügbar über seine Webseite, im Internet: http://cyber.law.harvard.edu/lessig (2.12.2000).

111 Vgl. **Lessig 1999**: «In the beginning, there were very few applications on the net. The net was no more than a protocol for exchanging data [...]. Once a protocol was specified, programmers could build programs that utilized it. Much of the software implementing these protocols was "open," at least initially − that is, the source code for the software was available with the object code. This openness was responsible for much of the early Net's growth.» (S. 102f.).

«1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty [...];

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions: [...]»[112]

Eine zunehmende Digitalisierung der Wissensdarstellung und der Transportmedien unterstützt den Wissenstransfer in besonderem Maße, da die Vervielfältigungs- und Transferkosten gegen Null tendieren und die Reichwerte der Informationsangebote durch verstärkte Nutzung des Internets "explodierend" zunimmt.[113] Dadurch können immer mehr Menschen an diesem Prozess des Wissenstransfers teilhaben.[114]

Wissen in seinen verschiedenen Erscheinungsformen (Software, Texte, Bilder, Formeln, Algorithmen etc.) wird innerhalb der "Open Source"-Bewegung nicht nach einem fiktiven Eigentumswert, sondern nach seinem realen Gebrauchswert beurteilt. Der Gebrauch des Wissens wird als ein Mittel zur Steigerung seines Gebrauchswertes angesehen, indem die Erfahrungen aus dem Gebrauch an die Wissensanbieter und andere Nutzer zurückfließen: Positive Erfahrungen bestätigen dabei den Anbieter in seinen Anstrengungen zur Innovation, negative Erfahrungen werden mit dem Ziel der Verbesserung des Angebots ausgewertet.

Besondere Bedeutung hat dieser Wissenstransfer in Bezug auf Belange der IT-Sicherheit, wie in Kapitel 6 erläutert wird.

Der Gebrauch des auf dem verbesserten Wissen aufgebauten Produkts, z.B. der Software, führt im Gegenzug auf Seiten des Nutzers zu positiven Erfahrungen, d.h. der Rechtfertigung seiner Bemühungen beim Testen und Melden der Fehler. Hinzu kommt, dass die Anbieter in der Regel auch

---

112 **The GNU General Public License, Version 2, June 1991**.

113 Vgl. **Evans/Wurster 2000**, S. 37: «Die explosionsartige Entwicklung der Konnektivität und die Einführung allgemein gültiger Standards [...] mit deren Hilfe jeder mit jedem und praktisch zum Nulltarif kommunizieren kann [...].» Die Autoren beschreiben die Auswirkungen der massiven Einführung des Internet als «*Aufbrechen des Kompromisses zwischen Reichhaltigkeit und Reichweite*» (S. 36).

114 Es wird berechtigterweise immer wieder daraufhingewiesen, dass die Teilhabe an den modernen Kommunikationsmitteln großen Teilen der Weltbevölkerung aus ökonomischen Gründen nicht möglich ist. Darin liegt die große Gefahr der vollständigen Abkopplung vieler Volkswirtschaften von der Weltwirtschaft. Lizenzkostenfreie "Open Source"-Alternativen könnten einen wichtigen Beitrag dazu leisten, diese Gefahr zu bannen.

selbst Nutzer und viele Nutzer − in geringerem Umfang − auch Anbieter sind, also ein ureigenes Interesse an einer Kontinuität der geschilderten Prozesse haben. Auf diese Art und Weise führt der Zyklus des Informationsaustausches zur stetigen Verbesserung und Erweiterung des Angebots, wovon alle Nutzer gleichermaßen profitieren.

Die hier geschilderten Prozesse zeigen die "Open Source"-Bewegung als eine klassische Ausprägungsform der von den Ökonomen für die Netzwerkökonomie beschriebenen Modelle[115] mit ihren "demand side driven" "positive feedback cycles".[116]

Zu beachten ist auch die enge Analogie der "Open Source"-Gemeinschaft zur Wissenschaftsgemeinschaft.[117] Genau betrachtet, handelt es sich nicht nur um eine Analogie, sondern um eine "Verzahnung" der beiden Gemeinschaften: Universitäten und Hochschulen gehören nicht nur zu den größten Anwendern von "Open Source"-Software, sie steuern auch umfangreiche Beiträge aus Forschungsprojekten und Studien- und Diplomarbeiten bei. "Open Source" bildet dabei einen integralen Bestandteil der universitären Grundlagenforschung im IT-Bereich und zunehmend auch in anderen Bereichen. Der mikro- und makroökonomische Nutzen dieses Know-how-Transfers ist allem Anschein nach bisher noch nicht untersucht worden.

Neuere ökonomische Forschungen insbesondere zu Fragen des ökonomischen Nutzens von scheinbar uneigennützigem Verhalten könnten eventuell weitere Erklärungen liefern.[118]

4.2.1. "Open Source"−Software

«We believe in rough consensus and running code.»[119]

In Form der "Open Source"-Software zeigt das oben formulierte "Open Source"-Paradigma die wohl bekannteste Ausprägung. "Ungehinderter Wissenstransfer" ist bei Software gleichbedeutend mit ungehindertem

- Quelltextransfer;
- Dokumentationstransfer;

---

115 Vgl. **Shapiro/Varian 1999, Turner 2000**.

116 Vgl. auch **Ardal 2000** und seine Untersuchung der Linux-Entwicklung.

117 Vgl. **Ardal 2000**, S. 93.

118 Vgl. z.B. **Lerner/Tirole 2000**, S. 2; **Ostrom 1999**.

119 Viele "Open Source"-Entwickler halten das Motto der IETF − "We believe in rough consensus and running code" − für ein gutes Leitbild auf dem Weg zu besserer Software. Vgl. **Bradner 1999**. Zu den Grundzügen der "Open Source"-Software-Entwicklung siehe insbesondere den Aufsatz "The Cathedral and the Bazaar" von *Eric S. Raymond* in **Raymond 1999,** S. 27ff.

- Lizenztransfer;
- Erfahrungstransfer.

Dem entspricht die "Open Source Definition" mit ihren wichtigsten Forderungen.

Es gibt weitere Bedingungen, wie z.B. die Verbote, Lizenzgebühren zu kassieren, bestimmte Anwender zu diskriminieren oder auch den Einsatz nur zu bestimmten Zwecken festzulegen.[120] Um einen Missbrauch der Bezeichnung "Open Source" zu verhindern, ist diese selbst rechtlich geschützt. Ein Gebrauch des Terminus' im Sinne der "Open Source"-Bewegung wird in der Praxis ohne Lizenzierung toleriert.

Eine Privatisierung der Software, d.h. des Wissens, etwa durch Exklusivnutzungsklauseln in den Lizenzen, würde die genannten Bedingungen verletzen und den Entzug der Lizenz (zur Nutzung der Bezeichnung "Open Source") verursachen. Rechtliche Schritte gegen die Lizenzverletzung könnten folgen. Eine Monopolbildung auf der Basis von "geistigem Eigentum" ist bei "Open Source"-Software *rechtlich* ausgeschlossen. Dadurch ist der Wettbewerb *innerhalb* des Software-Marktes und nicht *um* den Software-Markt zu führen.[121]

"Open Source"-Software wird als Quellcode (oder zusammen mit diesem[122]) mit Dokumentation und Beratung[123] bereitgestellt. Die Dokumentation erfolgt sowohl unmittelbar im Quellcode als auch mittelbar in Handbüchern, Webarchiven etc.

Zur Veranschaulichung dieser Aussage wird an dieser Stelle ein Ausschnitt aus dem Quellcode des Linux-Betriebssystemkerns (Version. 2.2.16, Datei /usr/src/linux/drivers/block/ataflop.c) abgebildet:

---

120 Sehr wohl aber dürfen Gebühren für die Zur-Verfügung-Stellung auf CD-ROM's als Aufwandsvergütung erhoben werden. Auf dieser Basis finanzieren die "Open Source"-Software-Distributoren ihr Angebot. Vgl. Open Source Definition im Internet: http://www.opensource.org/osd.html (3.12.2000).

121 Vgl. auch **Ardal 2000**, S. 43f., der das am Beispiel von "Linux" untersucht hat.

122 Die Software darf ohne Quellcode vertrieben werden, wenn dieser anderweitig leicht zugängig ist, z.B. im Internet. Vgl. Open Source Definition Nr.2, im Internet: http://www.opensource.org/osd.html (3.12.2000).

123 Die Beratung erfolgt typischerweise online, in Form von schriftlichen Frage-/Antwort--Dokumenten (FAQ's = Frequently Asked Questions) oder von Diskussionen − in Newsgroups oder direkt mit den Entwicklern und anderen Anwendern − per email.

```
/*
 *  drivers/block/ataflop.c
 *
 *  Copyright (C) 1993  Greg Harp
 *  Atari Support by Bjoern Brauel, Roman Hodek
 *
 *  Big cleanup Sep 11..14 1994 Roman Hodek:
 *   - Driver now works interrupt driven
 *   - Support for two drives; should work, but I cannot test that :-(
 *   - Reading is done in whole tracks and buffered to speed up things
 *   - Disk change detection and drive deselecting after motor-off
 *     similar to TOS
 *   - Autodetection of disk format (DD/HD); untested yet, because I
 *     don't have an HD drive :-(
 *
 *  Fixes Nov 13 1994 Martin Schaller:
 *   - Autodetection works now
 *   - Support for 5 1/4'' disks
 *   - Removed drive type (unknown on atari)
 *   - Do seeks with 8 Mhz
 *
 *  Changes by Andreas Schwab:
 *   - After errors in multiple read mode try again reading single sectors
 *  (Feb 1995):
 *   - Clean up error handling
 *   - Set blk_size for proper size checking
 *   - Initialize track register when testing presence of floppy
 *   - Implement some ioctl's
 *
 *  Changes by Torsten Lang:
 *   - When probing the floppies we should add the FDCCMDADD_H flag since
 *     the FDC will otherwise wait forever when no disk is inserted...
 *
 * ++ Freddi Aschwanden (fa) 20.9.95 fixes for medusa:
 *  - MFPDELAY() after each FDC access -> atari
 *  - more/other disk formats
 *  - DMA to the block buffer directly if we have a 32bit DMA
 *  - for medusa, the step rate is always 3ms
 *  - on medusa, use only cache_push()
 * Roman:
 *  - Make disk format numbering independent from minors
 *  - Let user set max. supported drive type (speeds up format
 *    detection, saves buffer space)
 *
 * Roman 10/15/95:
 *  - implement some more ioctls
 *  - disk formatting
 *
 * Andreas 95/12/12:
 *  - increase gap size at start of track for HD/ED disks
 *
 * Michael (MSch) 11/07/96:
 *  - implemented FDSETPRM and FDDEFPRM ioctl
 *
 * Andreas (97/03/19):
 *  - implemented missing BLK* ioctls
 *
 *  Things left to do:
 *   - Formatting
 *   - Maybe a better strategy for disk change detection (does anyone
 *     know one?)
 */

#include <linux/module.h>
....
static int MaxSectors[] = {
   11, 22, 44
};
static int BufferSize[] = {
   15*512, 30*512, 60*512
};

#define    BUFFER_SIZE    (BufferSize[DriveType])
```

```
unsigned char *DMABuffer;                  /* buffer for writes */
static unsigned long PhysDMABuffer;   /* physical address */

static int UseTrackbuffer = -1;           /* Do track buffering? */
MODULE_PARM(UseTrackbuffer, "i");

unsigned char *TrackBuffer;                       /* buffer for reads */
static unsigned long PhysTrackBuffer; /* physical address */
static int BufferDrive, BufferSide, BufferTrack;
static int read_track;          /* non zero if we are reading whole
tracks */

#define    SECTOR_BUFFER(sec)    (TrackBuffer + ((sec)-1)*512)
#define    IS_BUFFERED(drive,side,track) \
    (BufferDrive == (drive) && BufferSide == (side) && BufferTrack ==
(track))
...
```

Es handelt sich um einen Ausschnitt aus einem sogenannten "Treiber", Software zur Ansteuerung von Peripheriegeräten. Im konkreten Fall dient der Treiber der Ansteuerung eines Diskettenlaufwerks für Atari-Computer. Man erkennt die Kommentare der verschiedenen Entwickler über die von ihnen geleistete Arbeit. Man erkennt auch, dass die Entwickler namentlich bekannt sind. Bei Problemen mit dem Code wäre es also möglich − und ist es in der Praxis üblich −, die Entwickler zu kontaktieren und sie auf die Probleme hinzuweisen. Diese könnten dann ggf. das Problem analysieren und Fehler beseitigen.

Auch dem Laien dürfte bei Durchsicht des Quelltextausschnittes klar werden, dass der Quelltext nicht nur Gegenstand, sondern auch Mittel der Kommunikation zwischen Anwendern und Entwicklern im "Open Source"-Prozess ist. In diesem Punkt unterscheidet sich "Open Source"-Software wesentlich von proprietärer Software, bei der Quelltext kein Mittel der Kommunikation zwischen Anwendern und Entwicklern ist.

Software als "Erfahrungsgut" mit hohem Evolutionspotential eignet sich besonders gut, um den Feedback der Anwender in kürzester Zeit zu berücksichtigen. Linux als "Open Source"-Betriebssystem demonstriert in hervorragender Weise die Effektivität der Wirkmechanismen von Netzwerkökonomien.[124]

4.2.2. Freie Software versus "Open Source"−Software

> «The Free Software Movement has a more far-reaching and deeper goal: to give users the freedom to cooperate and participate in a community together. To have freedom, we must use software that respects our freedom. A non-free program takes away its users' freedom, so the only way to be free if you use a computer is to keep the non-free software off your computer.»[125]

---

124 Die bisher ausführlichste Darstellung dazu findet sich in **Ardal 2000**.

Die Bezeichnung "Open Source"-Software ist relativ neu,[126] nicht jedoch das "Open Source"--Paradigma, wie es oben formuliert ist. Wie erwähnt haben vergleichbare Ansätze zur Software-Entwicklung eine wesentlich längere Geschichte. Besonders zu betonen ist dabei das Wirken des Programmierers *Richard Stallman*, Gründungsmitglied der Free Software Foundation[127] und des GNU-Projekts. Unter der Bezeichnung "Free Software" entwickelte *Stallman* als Reaktion auf die zunehmende Durchsetzung von exklusiven Urheberrechtsansprüchen auf Software durch Unternehmen die Grundkonzepte der freien Weitergabe des Quellcodes von Software und die "General Public License" (GPL) als urheberrechtlich relevante Ausdrucksform seiner Auffassungen.

Die "Free Software"-Bewegung um *Richard Stallman* betont im Unterschied zu den Verfechtern der "Open Source"-Software-Idee die soziale Bedeutung der freien Verfügbarkeit von Software[128] durch den freiwilligen Verzicht auf urheberrechtliche Exklusivnutzung, wie sie bei proprietärer Software der Normalfall ist.

Die Freiheit, die der Anwender von "Free Software" besitzt, definiert *Richard Stallman* in vier Punkten:[129]

- die Freiheit, das Programm − egal zu welchem Zweck − auszuführen
- die Freiheit, das Programm den eigenen Bedürfnissen anzupassen, wozu der Zugriff auf den Quelltext unabdingbar ist
- die Freiheit, Kopien − gratis oder gegen Gebühr − weiterzuverteilen
- die Freiheit, bearbeitete Versionen zu verbreiten, sodass andere («the community») von den Erweiterungen profitieren können

Die vier von *Stallman* genannten Freiheiten korrespondieren nahezu exakt mit den Handlungs- und Verfügungsrechten der "Property Rights"-Theorie der neuen Institutionenökonomik, wie sie von *Nüttgens* und *Tesei* zusammengefasst wurden. Die Autoren unterschieden:[130]

- *Usus*: das Recht, ein Gut auf unterschiedliche Art zu nutzen
- *Abusus*: das Recht, ein Gut formal und materiell zu verändern
- *Übertragungsrecht*: das Recht, ein Gut frei zu veräußern und die Erlöse einzubehalten
- *Usus fructus*: das Recht, die aus der Nutzung eines Gutes entstehenden Gewinne zu erhalten bzw. die Pflicht, die aus der Nutzung eines Gutes entstehenden Verluste zu tragen

---

125 **Stallman 2000 a**.

126 Vgl. **Raymond 1999**, p. 205ff.

127 Die Free Software Foundation im Internet: http://www.fsf.org (3.12.2000).

128 Vgl. **Stallmann 1999** und die Darstellung des GNU-Projekts im Internet: http://www.gnu.org (29.11.2000).

129 Vgl. **Stallmann 1999**.

130 Vgl. **Nüttgens/Tesei 2000 a**, S. 10.

Die von *Richard Stallman* entwickelte "General Public License"[131] (GPL) sorgt für die rechtliche Implementierung dieser Freiheiten und Pflichten.

Stallman betont immer wieder, dass es nicht darum geht, kostenlose Software zu entwickeln. Es geht der "Free Software"-Bewegung *nicht* um einen Gegensatz von Software und Wirtschaft. Vielmehr geht es um einen Gegensatz von Software und Geschäftsmodellen, die Software als Ware wie ein beliebiges Industrieprodukt behandeln:

> «Since "free" refers to freedom, not to price, there is no contradiction between selling copies and free software. In fact, the freedom to sell copies is crucial: collections of free software sold on CD-ROMs are important for the community, and selling them is an important way to raise funds for free software development.»[132]

Das Ziel der "Free Software"-Bewegung ist es zu erreichen, dass alle Software "Free Software" mit den oben genannten Freiheiten ist. Das soll aber nicht die Abschaffung des Legalprinzips des geistigen Eigentums bedeuten, wie oft kolportiert wird, sondern die Schaffung einer nicht-proprietären Alternative (alternative Software mit alternativem Entwicklungs- und Vertriebsprozess). Die Anwender hätten dann die Wahl, sich für proprietäre Software zu entscheiden oder dagegen. In jedem Fall wäre es *ihre* Wahl − entsprechend *ihren* Bedürfnissen − auf dem Software-Markt eine Entscheidung zu treffen.

Die "Open Source"-Protagonisten betonen demgegenüber weniger eine soziale Haltung, sondern legen Wert auf den ingenieurtechnischen Charakter der Software-Entwicklung und den dafür notwendigen Informationsaustausch unter den Entwicklern und Anwendern. Sie treten für ein Nebeneinander von "Open Source"-Software und proprietärer Software ein. Viele ihrer Anhänger sehen proprietäre Software als Konkurrenten im Wettbewerb um das bessere Produkt an, woran es sich zu messen gilt.

---

131 Der Text der GPL über die Website der Free Software Foundation im Internet: http://www.fsf.org/ (3.12.2000).

132 Vgl. **Stallmann 1999**, S. 56.

4.2.3. Proprietäre Software

> «Der Lizenznehmer hat das auf die Anzahl der erworbenen Lizenzen beschränkte Recht der Nutzung und in Ausnahmefällen das Recht der Übertragung der erworbenen Lizenzen. Rechte auf Erträge entstehen dem Lizenznehmer nur im Rahmen der im Lizenzvertrag vereinbarten Nutzung der Software, die sich i.d.R. auf den unternehmensinternen Einsatz beschränkt. Aus dem Lizenzvertrag entsteht kein Recht auf die Veränderung von Software.»[133]

Proprietäre Software ist Software, deren Urheber sich exklusive Bearbeitungs-, Vervielfältigungs- und Weiterverbreitungsrechte vorbehält. Die Nutzung wird dem Lizenznehmer nur in engem Rahmen gestattet. Sie wird von Firmen, selten von Privatleuten, entwickelt und in der Regel zur Erzielung eines Unternehmensgewinnes aus Lizenzverkäufen vermarktet.

Dem Lizenznehmer wird nicht das Recht zur Bearbeitung oder Vervielfältigung oder Weiterverbreitung eingeräumt. Die Lizenz verbietet dem Lizenznehmer alle Aktivitäten, die über die bestimmungsgemäße Nutzung hinausgehen. Auch ein Recht zur Fehlerbeseitigung wird in der Regel nicht eingeräumt.

Um diese Aussagen zu veranschaulichen, wird an dieser Stelle ein Ausschnitt aus einer typischen Lizenz für proprietäre Software abgedruckt:[134]

> **« [...] You may not:**
> **a Modify, translate, reverse engineer, decompile, disassemble, create derivative Works based on, or copy (except for archival purposes) the program or the accompanying documentation [...]»**

Wie in diesem Ausschnitt gezeigt, verbieten proprietäre Lizenzen generell die Bearbeitung der Software, selbst wenn das Produkt (Vervielfältigungsexemplar mit Nutzungslizenz) rechtmäßig erworben wurde. Für analoge Werkskopien gibt es ein solches Bearbeitungverbot hingegen nicht. Der Erwerber eines Buches beispielsweise hat − für private Zwecke − das Recht, das gekaufte Buch in

---

133 **Nüttgens/Tesei 2000 c**, S. 11.

134 Es handelt sich um eine Lizenz des Netzwerkkomponentenherstellers Xircom, im Internet: http://www.xircom.com (25.10.2000). Dass wir einen Ausschnitt aus einer englischsprachigen Lizenz verwenden, trägt dem Umstand Rechnung, dass die meisten Lizenzen, mit denen ein Anwender konfrontiert wird, englischsprachig sind.

Abschnitte zu zerlegen, die Abschnitte anders anzuordnen oder auch Teile daraus zu entfernen bzw. darin einzufügen.

Die Ausübung des exklusiven Vervielfältigungsrechts und des Rechts zur Weiterverbreitung gestaltet sich naturgemäß etwas schwieriger: Digitale Werke lassen sich normalerweise verlustfrei vervielfältigen. Technische Maßnahmen wie Kopierschutz[135], Lizenzschlüssel[136] und Laufzeitbeschränkungen[137] ohne Produktregistrierung stellen gängige Lösungen zur Verhinderung der Herstellung großer Mengen von Kopien dar. Solche Kopierschutzmaßnahmen stehen in einem Spannungsverhältnis einerseits zu den gesetzlichen Ausnahmen, die im öffentlichen Interesse vorgesehen sind, und andererseits zu den Rechten, die ein Käufer mit dem Produktkauf − und Standardsoftware ist ein Fertigprodukt − erwirbt. Aus diesem Spannungsverhältnis resultieren regelmäßig Rechtsstreitigkeiten.

Proprietäre Software wird üblicherweise nicht mit Quellcode vertrieben, um den Anwendern keinen Einblick in das darin enthaltene Wissen zu ermöglichen und so dessen exklusiven Besitz zu wahren.[138] Stattdessen wird die Software als Binärcode, d.h. in nur für den Computer lesbarer Form vertrieben. Zur Veranschaulichung dieser Aussage wird an dieser Stelle ein Ausschnitt aus dem Code eines proprietären Programmes (Microsoft Windows 98, Second Edition) abgebildet:[139]

```
0600   0100   a526   c0b2   2000   6d61   6375   7365
7273   2e63   686d   0035   6700   00d0   8d06   0001
00a5   26c0   b220   006d   6473   2e63   686d   00b4
```

Moderne proprietäre Programme bestehen aus vielen Millionen solcher Zeilen.[140] Es ist offensichtlich, dass Software in dieser (proprietären) Form für Menschen unverständlich und nicht zur Bearbeitung oder zum Wissenstransfer gedacht ist. Auch der Fachmann kann in solcher Darstellung von Programmstrukturen nicht wesentlich mehr erkennen als ein beliebiger Anwender: Zahlenkolonnen.

---

135 Bsp.: Dongles, Kopplung an eine Kennziffer des Prozessors (CPUID), proprietäre Datenformate u.ä. kommen zum Einsatz.

136 Bsp.: Bei der Installation ist ein mitgelieferter Code (Schlüssel) einzugeben, ohne den die Software entweder gar nicht installiert wird oder nach der Installation nicht oder nur teilweise lauffähig ist.

137 Bsp.: Die Software wird vollständig installiert, ein Teil der Funktionalität steht aber nur zeitweise zur Verfügung. Nach Ablauf der Frist muss sich der Käufer des Produkts beim Anbieter registrieren lassen, um einen Lizenzschlüssel zu erhalten. Die Rechtmäßigkeit dieser Registrierungspflicht ist umstritten.

138 Zu den üblichen Lizenzbedingungen proprietärer Software siehe u.a. **Hoeren/Schumacher 2000**.

Es gab und gibt allerdings immer schon Ausnahmen, bei denen Unternehmen den Quellcode für ihre proprietäre Software mitlieferten. Es handelt sich dabei fast immer nicht um Standardsoftware.

139 Die Darstellung erfolgt in Hexadezimaldarstellung statt in der technisch äquivalenten Binärform.

140 Hatte Windows 3.1 im Jahr 1992 noch ca. 3 Mio. Zeilen Code, sind es bei Windows 2000 − 8 Jahre später − geschätzte 30 bis 60 Mio. Zeilen. Zahlen nach **Schneier 2000**, S. 357.

Die Fehlersuche in solchem Code ist praktisch unmöglich oder aber mit einem unvertretbar hohen Aufwand verbunden. Vertretbar ist der Aufwand eigentlich nur für den Besitzer des Quellcodes, d.h. den Anbieter der Software. Die einzige legale Möglichkeit der Fehlersuche besteht für einen Lizenznehmer darin, das Reaktionsverhalten der Software auf mehr oder weniger willkürliche Benutzereingaben und -interaktionen zu testen.[141] Dabei festgestellte Fehler können dem Hersteller gemeldet, dürfen und können jedoch nicht selbst korrigiert werden.

Eine effektive und legale Fehlerkorrektur kann nur bei Verfügbarkeit des Quellcodes vorgenommen werden und stellt eine Ausübung des Urheberrechts zur Bearbeitung des Werkes dar.[142] Die Aufgabe des Qualitätssicherung liegt bei proprietärer Software ganz beim Anbieter und das Vertrauen in die Sicherheit der Software ist einzig und allein auf den Aussagen des Anbieters aufzubauen, was bei geschäftskritischen Anwendungen als heikel einzustufen ist.

Gerade in jüngster Zeit haben Angriffe über das Internet auf Netzwerke großer Software-Hersteller deren Glaubwürdigkeit schweren Schaden zugefügt und das Vertrauen der Anwender in einseitige Beteuerungen der Hersteller abnehmen lassen. Erfahrungsgemäß wird jedoch nur ein Bruchteil der tatsächlich stattgefundenen Angriffe öffentlich bekannt, so dass die tatsächlichen Auswirkungen auf die Sicherheit proprietärer Software nur schwer abschätzbar sind.[143]

---

141 Eine solche Art von Test nennt man "Black Box"-Test: Man weiß nicht, was im Inneren des Systems (Programms) vor sich geht. Man kann nur feststellen, mit welchen Ausgaben es auf welche Eingaben reagiert. Vgl. **Köhntopp/Köhntopp/Pfitzmann 2000**.

142 UrhG §69c lit. 3.

143 Vgl. z.B. o.V.: **Bereitschaft zur Anzeige bei Computerdelikten nimmt ab.**

4.3. Entwicklungsmodelle für Software

Um eine gewisse Vorstellung von der Tätigkeit eines Software-Entwicklers zu vermitteln, sollen kurz die wichtigsten Schritte bei der Software-Entwicklung beschrieben werden. Diese unterscheiden sich bei proprietärer Software und "Open Source"-Software nicht. Wesentliche Unterschiede gibt es aber bezüglich der Aufgabenverteilung.

4.3.1. Die Arbeit eines Software−Entwicklers

Die folgende Grafik stellt ein Software-Entwicklungsmodell dar, wie es Studierenden der Informatik vermittelt wird:

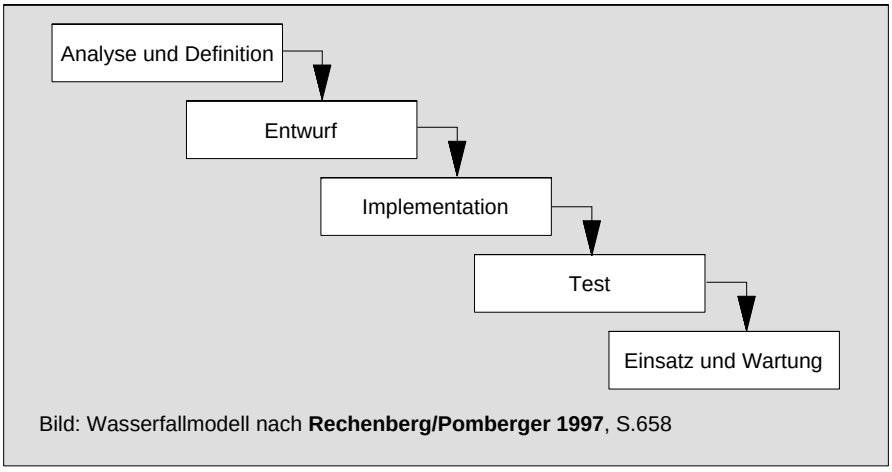


Bild: Wasserfallmodell nach **Rechenberg/Pomberger 1997**, S.658

Man sieht, dass Software-Entwicklung aus einer Reihe aufeinanderfolgender Schritte besteht. Bevor ein neuer Schritt begonnen wird, ist der aktuelle abzuschließen.

Solche sequentiellen Modelle haben den entschiedenen Nachteil, dass sie die tatsächlichen Abläufe bei der Software-Entwicklung falsch wiedergeben.[144] Um diesen Mangel auszugleichen, wurden dynamische Modelle entwickelt. Dazu gehören das Spiralmodell und zyklische Modelle.[145] Insbesondere die neueren zyklischen Modelle werden heutzutage genutzt, um den Prozess der Software-Entwicklung zu beschreiben. Im zyklischen Modell wird «*Software nicht mehr als ein Produkt, sondern als eine Folge von Versionen verstanden*»[146]:

---

144 Vgl. **Rechenberg/Pomberger 1997**, S. 658.

145 Vgl. **Rechenberg/Pomberger 1997**, S. 658f. In diesem Modell wird auch deutlich, was die Autoren von **Bessen/Maskin 2000** mit «*sequential innovation*» meinen: Die vielen kleinen Schritte auf dem Weg zu einer neuen Version.

146 **Rechenberg/Pomberger 1997**, S. 658.

> «Die Softwareentwicklung besteht dann aus einer Folge von Zyklen, in denen aufbauend auf die letzte Version eine neue Version hergestellt und eingesetzt wird. In diesem Modell sind Herstellung und Einsatz verschränkt, die Wartung entfällt und wird durch Pflege der aktuellen und Übergang zur nächsten Version ersetzt.»[147]

Überträgt man diese Modellvorstellungen in einen Zyklus von einzelnen Schritten, lässt sich dieser beispielsweise so darstellen:[148]

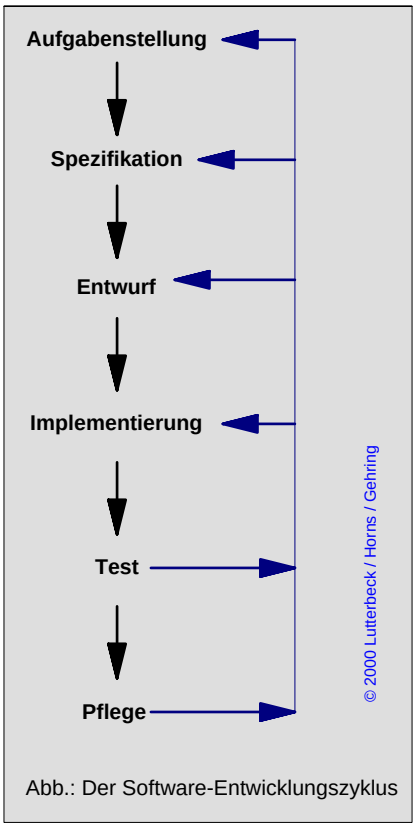


Abb.: Der Software-Entwicklungszyklus

Die Vorgehensweise nach diesem Modell[149] läßt sich folgendermaßen beschreiben:

(1) Ausgehend von einer Aufgaben- oder Problemstellung wird eine Spezifikation für eine mögliche Lösung erarbeitet.

---

147 **Rechenberg/Pomberger 1997**, S. 660.

148 Im Kern lässt sich die Tätigkeit des Software-Entwicklers mit dem hier gezeigten Modell beschreiben, das sich eher an **Vixie 1999** orientiert.
Im Internet: http//www.oreilly.com/catalog/opensources/book/vixie.html (07.11.2000).

149 Es gibt andere Modelle, wie z.B. das V-Modell für die öffentliche Verwaltung (vgl. **Stahlknecht 1993**, S. 238f.) oder das ältere Wasserfall-Modell (vgl. **Stahlknecht 1993**, S. 240ff; **Rechenberg/Pomberger 1997**, S. 657ff).

(2) Aus der Spezifikation wird ein Entwurf für das Systemdesign abgeleitet.

(3) Die Spezifikation wird implementiert.

(4) Die Implementierung wird getestet.

(5) Nach den erfolgreichen Tests wird die Software(-Version) in Betrieb genommen und muss gepflegt werden.

In Abhängigkeit von den Ergebnissen jedes Einzelschrittes kann es notwendig werden, einen der vorangegangenen Schritte modifiziert zu wiederholen. Sollte sich beispielsweise während der Implementierungsphase zeigen, dass der Entwurf fehlerhaft ist, muss ein neuer Entwurf angefertigt und in der Folge eventuell eine komplett neue Implementierung vorgenommen werden.

Die Entwicklungskosten im gezeigten Modell steigen immer dann stark an, wenn mehr als ein Schritt rückwärts gemacht werden muss, um eine lauffähige, stabile Version der Software zu erzeugen. Je früher im Entwicklungsprozess Fehler entdeckt und beseitigt werden können, desto kostengünstiger ist der Software-Entwicklungszyklus.

Dieses Modell ist relativ abstrakt und nicht besonders gut geeignet, die jeweiligen Spezifika von proprietärer Software-Entwicklung oder "Open Source"-Software-Entwicklung wiederzugeben. Diese zeigen sich vor allem in den organisatorischen Ausprägungen der einzelnen Schritte, d.h. den Antworten auf die Frage danach, *wer was wann* macht.

In den folgenden Abschnitten wird das abstrakte Modell sowohl für proprietäre Software-Entwicklung als auch für "Open Source"-Software-Entwicklung detaillierter angegeben.

4.3.2. "Wiederverwendung" als Paradigma

> «When experts work on a particular problem, it is unusual for them to tackle it by inventing a new solution that is completely distinct from existing ones. They often recall a similar problem they have already solved, and reuse the essence of its solution to solve the new problem. This kind of 'expert behaviour', the thinking in problem-solution pairs, is common to many different domains, such as architecture [Ale79], economics [Etz64] and software engineering [BJ94]. It is a natural way of coping with any kind of problem or social interaction [NS72].»[150]

Das Zitat stammt aus einem klassischen Lehrbuch für moderne Software-Entwicklung. "Reuse", wie die Autoren schreiben, kennzeichnet das Wesen von Expertenhandeln. Aus der modernen Soft-

150 **Buschmann et al. 1996**, S. 2; vgl. auch **Gamma et al. 1994**, S. 1: «One thing expert designers know not to do is solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past. When they find a good solution, they use it again and again. Such experience is part of what makes them experts.»

ware-Entwicklung ist dieses Grundprinzip der Wiederverwendung nicht mehr wegzudenken. Es nimmt die unterschiedlichsten Formen an:

- **Bibliotheken**, d.h. Sammlungen ablauffähiger Programmfunktionen[151]
- **Objekte**, d.h. die lauffähige Kombination aus Datenstrukturen und darauf operierenden Funktionen[152]
- **Patterns**, d.h. Beschreibungen von häufig wiederkehrenden Problemen und erprobte Ansätze zu deren Lösung[153]

Der Grund für die ubiquitäre Gegenwärtigkeit der Wiederverwendung ist darin zu sehen, dass

- Zeit gespart wird, um Code zu schreiben;
- Fehler, die bei anderen Projekten bereits gemacht wurden, nicht erneut gemacht werden;
- vorhandener Code die Erfahrung von Experten beinhaltet, die nicht ohne Weiteres beliebig reproduzierbar ist;[154]
- gepflegter Code zahlreiche Optimierungen enthält.

Nur durch die Wiederverwendung erprobter Ideen und bewährten Codes ist es überhaupt mit vertretbarem Aufwand, d.h. in *akzeptabler Zeit*, zu *akzeptablen Kosten*, mit *akzeptabler Fehlerquote*, möglich, komplexe und funktionsreiche Software zu schreiben. Die zyklischen Entwicklungsmodelle für Software reflektieren diese Tatsache und sprechen von Software-Versionen, anstelle von (irgendwann fertigen) Software-Produkten[155]. Jede neuere Version eines Programmes "erbt" Code von ihren Vorgängern und so die Erfahrungen der Entwickler.

Das Paradigma der Wiederverwendung ist sowohl für die Entwicklung proprietärer Software als auch nicht-proprietärer Software akzeptiert und wird breit angewandt. In der Art und Weise, wie Wiederverwendung praktiziert wird, unterscheidet sich die nicht-proprietäre Software-Entwicklung allerdings erheblich von der proprietären. Ein Blick in die jeweiligen Lizenzbedingungen offenbart, dass bei proprietärer Software Wiederverwendung grundsätzlich innerhalb des Unternehmens, bei nicht-proprietärer Software ("Free Software", "Open Source"-Software) dagegen unabhängig von Firmeninteressen praktiziert wird.

---

151 Vgl. **Holt/Morgan 1994**, S. 230.

152 Vgl. **Gamma et al. 1995**, S. 361.

153 Vgl. Alexander et al. 1977, zitiert bei **Gamma et al. 1995**, p. 2.

154 Vgl. **Raymond 1999**, S. 33: «Good programmers know what to write. Great ones know what to rewrite (and reuse).»

155 Die Begrifflichkeit von "Produkt" im Kontext der Software-Entwicklung ist eine andere als aus der Perspektive kaufmännischer Tätigkeiten. Gemeint ist eine (entsprechend einer vorgegebenen Spezifikation) ganz und gar vollständige Software.

4.3.3. Die Entwicklung proprietärer Software im Unternehmen

Verfeinert man das abstrakte Modell um die Aufgabenverteilung bei der Entwicklung proprietärer Software, so erhält man in etwa folgendes Bild:

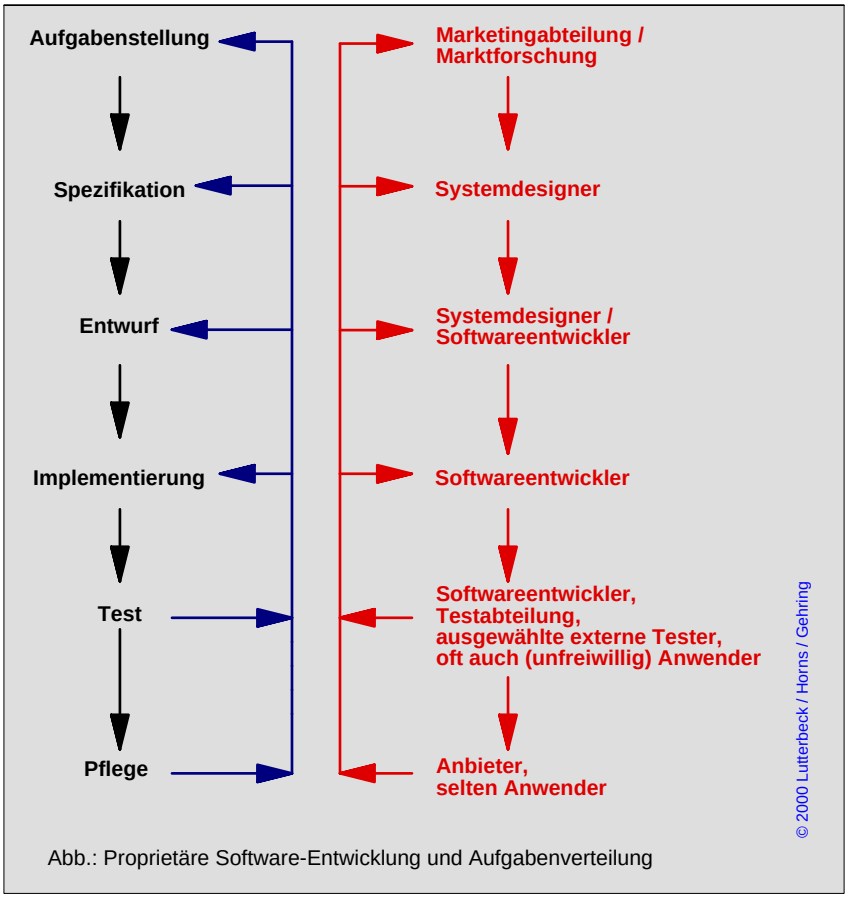


Abb.: Proprietäre Software-Entwicklung und Aufgabenverteilung

Ausgangspunkt für eine Produktentwicklung ist in der Regel die unternehmensinterne Abteilung für Marketing und/oder Marktforschung. Dort wird abgeschätzt, was sich als Software-Produkt verkaufen lässt. Dafür spielen Kundenwünsche, aber auch die Verfügbarkeit der Entwicklungsressourcen und die Unternehmensstrategie eine Rolle. Im Ergebnis der Untersuchungen und Planungen entsteht ein Konzept für ein neues Software-Produkt oder für eine neue Version eines bestehenden Produkts.

Aus dem Marketing-/Produkt-Konzept erarbeiten die Systemdesigner das Design für das Produkt, wobei die wesentlichen Elemente in abstrakten Einheiten, Modulen und Interaktionen zwischen diesen Einheiten abgebildet werden. Die zukünftigen Kunden, d.h. die potentiellen Anwender, werden in diesen Prozess nicht weiter einbezogen. Sonderwünsche finden natürlich keine Berücksichtigung.

Aus dem Grobdesign erarbeiten die Systemdesigner in Absprache mit den Software-Entwicklern den konkreten Entwurf der Software, oft in Form eines Prototyps.[156] Wurden für das Systemdesign unterstützende Software-Werkzeuge benutzt, so wurde das Codegerüst für den Prototypen

156 **Rechenberg/Pomberger 1997**, S. 660: «Ein Prototyp ist eine spezielle Ausprägung eines ablauffähigen Softwaresystems.»

automatisch generiert, d.h. den Software-Entwicklern verbleibt im Wesentlichen die Aufgabe, das Codegerüst entsprechend der geforderten Funktionalität auszubauen. Die zukünftigen Anwender werden auf dieser Stufe in der Regel nicht einbezogen.

Es soll noch einmal betont werden, dass sich die hier gemachten Ausführungen auf Standardsoftware im Massengeschäft beziehen. Im Projektgeschäft mit kundenspezifischer Software (Individualsoftware) werden die Anwender oft stärker in den Entwicklungsprozess integriert als hier dargestellt.

Während und nach der Implementierung testen die Software-Entwickler den von ihnen geschriebenen Code. Sollte das Unternehmen über eine spezielle Testabteilung verfügen, übernehmen deren Mitarbeiter große Teile des systematischen Testens. Dafür kommen zum Teil automatische Werkzeuge zum Einsatz, die eine − vorzugebene − Funktionalität testen. Getestet wird mithin, ob die Software sich entsprechend ihren Spezifikationen verhält. Anwender werden in dieser frühen Testphase üblicherweise nicht einbezogen.

Unmittelbar nach der Implementierung weist Software häufig viele Fehler auf. Im Testlauf werden die Fehler nach und nach bestimmt und durch Änderungen in der Codebasis beseitigt. Schwere Fehler oder sich ändernde Anforderungen machen ggf. eine Re-Implementierung und/oder ein Re-Design nötig.

Nach etlichen Durchläufen des Codieren/Testen-Zyklus' erreicht das Produkt, d.h. die Software, den sogenannten Beta-Status. An dieser Stelle kommt ein Kreis ausgewählter oder freiwilliger Anwender als Beta-Tester zum Einsatz. Sie erhalten das unfertige Produkt und testen es. Ihre Erfahrungen mit dem Produkt melden sie an den Hersteller zurück, wo die Entwickler wiederum Modifikationen am Code vornehmen und diesetTesten. Markant ist die klare Trennung zwischen Entwicklern und Anwendern (externen Testern).

Wenn so eine − aus Unternehmenssicht − ausreichende Produktqualität erreicht worden ist, wird die Software vermarktet. Jetzt zeigt sich, ob die von der Marktforschung antizipierten Kundenwünsche durch das Produkt befriedigt werden. Wenn ja, dann wird es gekauft werden. Wenn nicht, dann wird das Produkt nur dann gekauft werden, wenn dem Kunden keine andere Wahl bleibt.

Nach dem In-Verkehr-Bringen der Software und bei ihrem Einsatz tauchen oft noch Fehler oder Sicherheitslücken auf, die eine Wartung erfordern. Dazu bieten die Hersteller dann Updates, Patches, Service Packs, etc. an, die der Kunde selbst zu beschaffen und zu applizieren hat.

Funktionserweiterungen werden in Form von Upgrades, neuen Releases, neuen Editionen etc. regelmäßig separat vermarktet.

Der Einfluss der Kunden und Anwender auf den Prozess der Produktentwicklung ist insgesamt nicht sehr groß. Sonderwünsche finden praktisch keine Berücksichtigung.[157] Entscheidend für die Produkteigenschaften sind vor allem unternehmensinterne Maßgaben, z.B. die Forderung nach

---

157 Man sollte sich an dieser Stelle klar machen, dass schon der Wunsch eines kleineren oder ärmeren Staates nach einer Sprachanpassung einer bestimmten Standardsoftware aus der Sicht eines großen Software-Konzerns als Sonderwunsch zu gelten hat.

Kompatibilität mit älteren Produkten aus dem eigenen Hause oder die Schwächung der Marktposition von Konkurrenzangeboten.

4.3.4. Die Entwicklung von "Open Source"−Software

> «Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.»[158]

Die Erweiterung des abstrakten Modells um die Aufgabenverteilung bei der Entwicklung von Software im "Open Source"-Prozess, führt zu folgendem Bild:

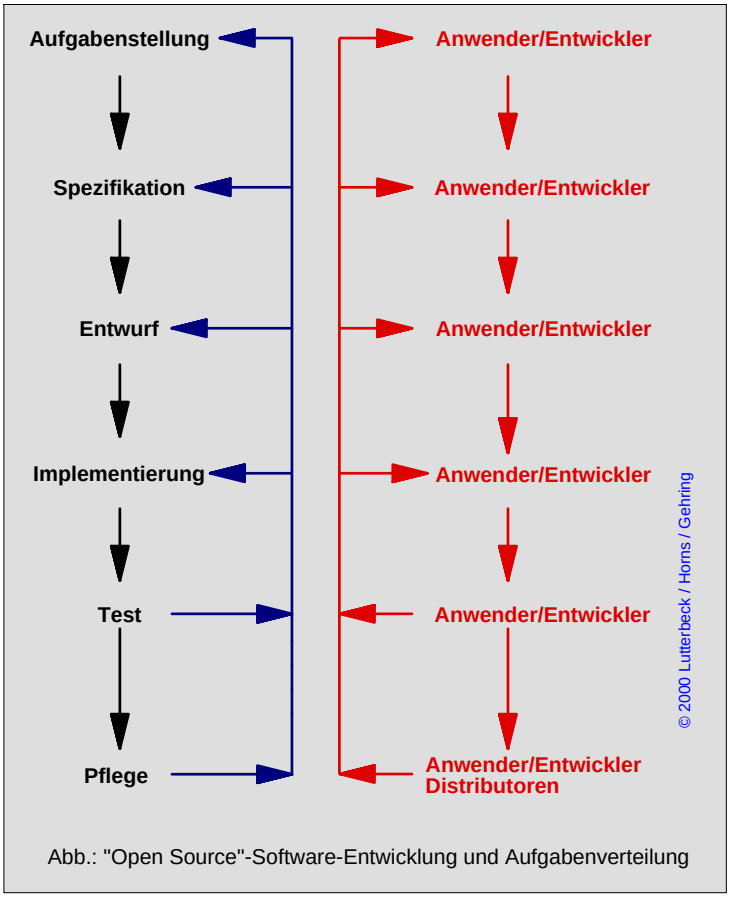


Abb.: "Open Source"-Software-Entwicklung und Aufgabenverteilung

Der Ausgangspunkt der "Open Source"-Software-Entwicklung ist nicht die Marketingabteilung einer Software-Firma. Aus Literatur und der Praxis sind vielfältige Ausgangspunkte bekannt, deren Gemeinsames *Raymond* folgendermaßen identifiziert:

158 **Raymond 1999**, S. 37.

> «Every good work of software starts by scratching a developer's personal itch.»[159]

Das soll besagen, dass bei "Open Source"-Software am Anfang die Entscheidung eines Entwicklers steht, Software für einen bestimmten Zweck zu schreiben. Nicht organisatorische oder finanzielle Randbedingungen initiieren demnach "Open Source"-Software-Projekte, sondern persönliche Interessen.

Es gibt eine Anzahl von "Open Source"-Software-Projekten, die das belegen. Für viele Software-Projekte kann die Aussage in ihrer Pauschalität jedoch keine befriedigende Erklärung liefern. Zumal durch die zunehmende Akzeptanz von "Open Source"-Software in Unternehmen Randbedingungen hinzutreten, die eher denen in klassischen Software-Unternehmen gleichen als dem Bild vom "einsamen Hacker". Damit werden Fragen nach der Motivation aufgeworfen, die sich nicht so einfach beantworten lassen. Andere Ansätze zum "Open Source"-Projektmanagement haben begonnen, Entwickler und potentielle Auftraggeber über eine Internetplattform zusammenzubringen.[160] Jedenfalls ist das Bild alles andere als einheitlich. Hier besteht Bedarf an weiterer Forschung.

Wird ein "Open Source"-Software-Projekt begonnen, steht es wie jedes andere Software-Projekt vor dem Problem der Spezifikation: Was soll die Software machen und wie soll die Funktionalität implementiert werden? Ein Teil der Spezifikation wird durch Rückgriff auf "Offene Standards" formuliert. Das hat verschiedene Vorteile für Entwickler und Anwender:

- Es existieren in der Regel Referenzimplementierungen, deren Code wiederverwendet werden kann.
- Die Überprüfbarkeit durch unabhängige Fachleute ist gegeben. "Offene Standards" werden in der Regel gründlich geprüft, bevor sie Geltung erlangen. Viele Spezifikationsfehler können so vermieden werden.
- Die Interoperabilität zu anderer Software, insofern diese ebenfalls "Offene Standards" implementiert, wird bereits in der Spezifikation berücksichtigt.

Ein anderer Teil der Spezifikation wird − in unterschiedlichster Form − skizziert und im Internet veröffentlicht.[161] Dort kann sie von Interessenten − potentiellen Anwendern und Entwicklern − in aller Welt begutachtet und kritisiert werden. Berücksichtigt der Entwickler ernstzunehmende Kritik, entsteht eine verbesserte Spezifikation.

---

159 **Raymond 1999**, S. 32.

160 Siehe: http://www.sourceforge.net (30.11.2000) oder http://collab.net (30.11.2000).

161 Das wohl einflussreichste "Open Source"-Projekt − das Linux-Betriebssystem − wurde durch eine Nachricht in einer Internet-Newsgroup durch *Linus Torvalds* ins Leben gerufen. Vgl. **Torvalds 1999**.

Hat der Entwickler der Spezifikation Grundregeln der Software-Entwicklung beachtet, lassen sich aus der Spezifikation einzelne Aufgaben zur Implementierung von Code-Modulen ableiten. Entspricht das Projekt den Interessen und/oder Bedürfnissen von Entwicklern, beginnen diese − (weltweit) verteilt und über das Internet kommunizierend − mit der Implementierung der Code-Modulen und testen deren Funktionalität. Der dabei entstehende Code wird in der Regel zeitnah im Internet veröffentlicht.[162]

Der öffentlich zugängliche Code wird nicht nur von den Entwicklern selbst, sondern auch von potentiellen Anwendern getestet und begutachtet. Fehlermeldungen und Kritik werden − über unterschiedliche Kanäle − direkt an die Entwickler gerichtet. Diese können darauf reagieren und ggf. den Code anpassen. Bei größeren Projekten finden sich oft Anwender bereit, kleinere Code-Korrekturen − Patches[163] − oder zusätzliche Modulen beizusteuern und dergestalt die Code-Basis zu verbessern. Charakteristisch ist, dass die Motivation der Entwickler das treibende Moment ist.

Stück für Stück entsteht so in weltweiter Kleinarbeit ein Programm. Von Zeit zu Zeit werden lauffähige Versionen erzeugt und zum Testen (und Verbessern) veröffentlicht. Durch die Code-Verfügbarkeit im Internet können Interessenten aber auch ganz nach Belieben selbst den Quellcode compilieren und Testversionen der Software erzeugen. Der ganze Prozess wird durch die Bedürfnisse der Entwickler und (Test-)Anwender gesteuert, nicht durch den Zeitplan einer Firma.[164]

4.4. Organisation, Kooperation und Kommunikation

Der folgende Abschnitt reißt einige Aspekte an, die bisher nicht befriedigend behandelt wurden und soll vor allem dazu dienen, den Forschungsbedarf aufzuzeigen.

Organisation

> «Das Open Source Modell besitzt andere Bedingungen als die Publikumsgesellschaft, in dem es keine Eigentümer, keine Management und keine Mitarbeiter wie in einer Publikumsgesellschaft gibt.»[165]

Entwickler proprietärer Software sind in klassischen Unternehmen mit mehr oder weniger flachen Weisungshierarchien organisiert. Über das Für und Wider der einen oder anderen Form der

---

162 Z.B. auf Concurrent Version System (CVS)-Servern, die der Verwaltung der Codebasis dienen.

163 Der Apache-Webserver ist auf diese Weise aus dem NCSA-Server hervorgegangen. Den vielen Patches verdankt er auch seinen Namen: «a patchy server». Vgl. **Apache FAQ**, im Internet über: http://www.apache.org (30.11.2000).

164 Ausnahmen bestätigen auch hier die Regel. Inbesondere bei den Distributoren spielen Marketingüberlegungen eine wichtige Rolle.

165 **Nüttgens/Tesei 2000 c**, S. 10.

Unternehmensorganisation findet sich eine Unmenge an Literatur. Deren Diskussion reicht über den Rahmen dieses Gutachtens hinaus und kann außer Acht bleiben.

Die organisatorischen Hintergründe bei "Open Source"-Software sind ganz offensichtlich anderer Natur und nur wenig untersucht. "Communities", "Netzwerke", "komplexe Systeme" sind einige Stichwörter, unter denen die Materie in der wenigen vorhandenen Literatur behandelt wird.[166] Daraus lassen sich lediglich Indizien für die Vorteile einer Organisationsform entlang von Aufgabenbereichen anstelle von Eigentums- bzw. Dienstverhältnissen bei bestimmten Problemen, wie z.B. der Software-Entwicklung, ableiten.

> «[...] informelle Gruppen, die sich selbst organisieren, selbstgewählte Zwecke erfüllen und ihre Führung selbst bestimmen. Die Mitgliedschaft ist freiwillig, das heißt, jeder Beteiligte weiß tendenziell, wann und ob er mitmachen sollte. Jeder weiß zudem, ob er etwas beizutragen hat und ob er von seiner Teilnahme profitieren wird.»[167]

An die Natur der "Open Source"-Bewegung angelehnte Organisationsmodelle und -prozesse finden sich also auch in modernen Unternehmen. Die Auslassung im obenstehenden Zitat beginnt mit folgenden Worten: «Praktiker-Gemeinschaften sind». Im selben Text erläutern *Wenger* und *Snyder*, welche Voraussetzung zur Entstehung der von ihnen beschriebenen "Praktiker-Gemeinschaften" gegeben sein muss:

> «Gemeinschaften von Praktikern bilden sich in Unternehmen, die auf den Faktor Wissen setzen.»[168]

Es hat also den Anschein als würde die zunehmende Einbeziehung von (spezialisiertem) Wissen in die breite Produktion aller möglicher Güter − in diesem Fall "Open Source"-Produkte einbezogen − Implikationen für die Organisationsformen geben, in der diese Produktion abläuft. Diese starke Einbeziehung wurde erst durch den Einsatz modernster Kommunikationsmittel und der Speicherung von (Experten-)Wissen zum jederzeitigen Abruf möglich.

---

166 Vgl. **Nüttgens/Tesei 2000 a**; **Ardal 2000**.

167 **Wenger/Snyder 2000**, S. 56.

168 **Wenger/Snyder 2000**, S. 62.

> «The collaborative nature of open source software development has been hailed in the business and technical press as an important organizational innovation.»[169]

Das Internet liefert das beste Beispiel dafür, wie das Wissen heutzutage jederzeit an jeden Ort gelangen kann. Damit ändert sich aber auch der Umgang mit dem Wissen selbst und Organisationsformen, die auf der Exklusivität der Verfügbarkeit bestimmten Wissens aufgebaut sind, verlieren in der einen oder anderen Form ihr Fundament. Kommunikation kann bei fehlender Exklusivität nicht mehr erfolgreich per Weisung "von Oben nach Unten" erfolgen, sondern muss Formen des Dialogs annehmen. Das ist es wohl, worauf auch die Autoren des "Cluetrain-Manifesto"[170] hinweisen wollen.

Hat die Entwicklung proprietärer Software ihren Ausgangspunkt in betriebswirtschaftlichen Interessen, sind für die "Open Source"-Bewegung soziale Momente,[171] insbesondere die internationale Organisation der Kommunikationsbeziehungen auf der Basis des Internet, konstitutiv.

Weder die Eigentümerschaft an Unternehmenswerten noch die Gewinnerzielungsabsicht sind ausschlaggebende Kriterien für die "Open Souce"-Entwicklung. Eine Organisationsform entlang von Eigentums- oder Dienstverhältnissen in Bezug auf die Software wäre nicht herzustellen und wohl auch inadäquat.

Welches Potential das "Open Source"-Paradigma für die Organisation anderer Bereiche menschlicher Tätigkeit bietet, ist noch weitgehend unerforscht. Die Wissenschaft sollte sich dieser Aufgabe stellen.

Aufgabenteilung und Spezialisierung

Umfangreiche menschliche Tätigkeit wird durch Aufgabenteilung und Spezialisierung bewältigt. Die Software-Entwicklung bildet da keine Ausnahme. Die Spezialisierung bringt es jedoch − unter anderem − mit sich, dass in bestimmten Bereichen ein Mangel an geeigneten Fachkräften entstehen kann. Der aktuelle Arbeitskräftemangel im IT-Bereich macht das Problem deutlich. In der "Open Source"-Bewegung werden die Vorteile eines Ansatzes demonstriert, bei dem die Fachkräfte mit

---

169 Vgl. **Lerner/Tirole 2000**, S. 1., und auch **Turner 2000**, **Chamberlin 1998**, **Mowshowitz 1998**.

Nicht jeder IT-Fachmann kann sich indes mit diesen Entwicklungen, auch schon mal als *«Graswurzelaktivitäten»* bezeichnet (**Endres 2000**, S. 317), anfreunden. Hinter solchen Vorbehalten ist jedoch oft Unkenntnis der Grundlagen der "Open Source"-Bewegung auszumachen, vgl. z.B.: *«[...] wird dafür plädiert, Programme nur als sog. "freie" Software weiterzugeben, d.h. ohne Urheberrechts- oder Patentschutz in Anspruch zu nehmen.»* **(Endres 2000**, S. 316). Die fundamentale Bedeutung des Urheberschutzes für die Herausbildung und kontinuierliche Weiterentwicklung der "Open Source"-Bewegung ist dem Autor offensichtlich unbekannt.

170 **Locke 2000**.

171 Vgl. **Stallman 1999**.

ihrem Wissen nicht zu den Problemen transportiert werden, sondern umgekehrt, die Probleme zu den Fachkräften. *Raymond* formuliert diese Erkenntnis etwas salopp:

> «If you have the right attitude, interesting problems will find you.»[172]

Es bleibt zu untersuchen, in welchen Bereichen mit hohem Grad an Aufgabenteilung und Spezialisierung dieser Ansatz − "Probleme zu den Experten" statt "Experten zu den Problemen" − erfolgreicher sein kann als andere.

4.5. Zusammenfassung

> «It'll be an Open-Source World.»[173]

So lautet das Fazit, das *Delio* kürzlich aus einer Studie des Marktfoschungsunternehmens Forrester Research zog.[174] Ob sich diese Prognose bewahrheiten wird, muss hier dahingestellt bleiben. Einiges spricht dafür.

Ein Vergleich zwischen proprietärer Software und "Open Source"-Software wird zwangsläufig nur Einzelaspekte aus der Gesamtproblematik erfassen können. Zu vielfältig sind mittlerweile die Einsatzgebiete und Zusammenhänge, als dass sie in einem Kurzgutachten wie dem vorliegenden vollständig berücksichtigt werden können.

Die ökononomischen Schlussfolgerungen aus den Darstellungen in diesem Kapitel sind evident:

Die Offenheit des "Open Source"-Prozesses bringt es mit sich, dass die wichtigsten Ressourcen nicht im Besitz oder unter der Kontrolle eines einzelnen Unternehmens sind, mithin die Gefahr der Herausbildung eines Monopols (basierend auf geistigem Eigentum in Technologiebereichen) praktisch nicht besteht. Statt auf proprietäre Standards wird auf offene Standards gesetzt. So kann ein "Lock-in", d.h. die praktisch unauflösbare Koppelung eines Nutzers an eine bestimmte Technologie bzw. deren Anbieter, vermieden werden. Aus Wettbewerb *um den Markt* wird Wettbewerb *im*

---

172 **Raymond 1999**, S. 35.

173 **Delio 2000**.

174 Der Senior-Editor des "Linux Journal", *Doc Searls*, überschrieb seinen Eindruck nach dem Besuch der Linux Business Expo Anfang November so: «*World Domination? Heh.*», **Searls 2000**.

Markt.[175]

Die weite Verbreitung der "Open Source"-Produkte (Software, Dokumentation, Content usw.) nutzt Netzwerkexternalitäten[176] optimal aus und bewirkt maximale (Nutz-)Wertsteigerungen: "Open Source"-Software ist für die Anwender dann am wertvollsten, wenn es möglichst viele Anwender gibt. Die schnellstmögliche Verbreitung wird dabei erreicht, wenn die Ein- oder Umstiegskosten für die potentiellen Anwender am geringsten sind. Diese Kosten streben gegen ein Minimum, wenn einerseits die Entwicklungskosten auf möglichst viele Träger verteilt werden und es andererseits viele konkurrierende Anbieter (Distributoren) gibt, die vergleichbare Produkte zu möglichst geringen Kosten anbieten. Die weit verteilten und deshalb pro Nutzer niedrigen Investitionskosten senken dabei potentielle Marktzutrittsbarrieren. Genau dieser Fall ist bei "Open Source"-Produkten gegeben.

Aus ökonomischer Perspektive zeigt der "Open Source"-Markt Merkmale einer *vollständigen Konkurrenz* ("perfect competition") und entspricht damit stärker einem marktwirtschaftlichen Leitbild als der klassische Softwaremarkt mit seiner Tendenz zur Bildung (bereichsspezifischer) natürlicher Monopole.[177]

Dies ist auch das Ergebnis der Untersuchung von *Nüttgens* und *Tesei*: Die sinkenden Transaktionskosten geben Anreize, dieses Modell der Entwicklung und des Vertriebs von Software zu bevorzugen.

> «Offene Systeme berauben Anbieter ihrer Monopolgewinne und sorgen für mehr Wettbewerb auf Gebieten, wo Standards gelten. Um weiterhin profitabel zu bleiben, müssen die Hersteller Nischenprodukte herausbringen, die sich in die Architektur offener Systeme einfügen − und sie müssen sich um eine verbesserte Preis-Leistungs-Relation bemühen. Immerhin kommt es den Herstellern entgegen, daß so viele Anwender mit und in diesen Systemen arbeiten werden und daraus ein weit größeres Marktpotential entsteht.»[178]

---

175 Vgl. **Shapiro/Varian 1999**, S. 228f.

176 Zu *Netzwerkexternalitäten* vgl. die Kommentierung von **Liebowitz/Margolis 1998**. Die Autoren weisen zu Recht darauf hin, dass von Netzwerkexternalitäten in Bezug auf Netzwerkeffekte nur gesprochen werden kann, solange diese nicht, z.B. durch den Netzwerkbesitzer, internalisiert wurden. Bei "Open Source" kann man demnach berechtigt von Netzwerkexternalitäten sprechen.

177 Vgl. u.a.: **Shapiro/Varian 1999**, S. 183ff; **Lipsey/Chrystal 1999**, S. 165; **Ardal 2000**, S. 17ff; **Schmidt 1999**, S. 5ff, **Nüttgens/Tesei 2000 c**.

178 **Benjamin/Blunt 1993**.

Das voranstehende Zitat stammt aus einer Prognose von *Robert J. Benjamin* und *Jon Blunt* aus dem Jahre 1993(!) für die 90er Jahre bis zum Jahr 2000. Betrachtet man die Entwicklung der vergangenen Jahre, so erstaunt die Hellsichtigkeit der Autoren. Zwar hat sich ihre Voraussage nicht in der angenommenen Größenordnung verwirklicht, aber in der Tendenz: "Open Source"-Software realisiert offene Systeme und der Marktanteil des größten Software-Herstellers der Welt − nach richterlicher Feststellung Monopolist − sinken. Im Markt für PC-Betriebssysteme kommt Wettbewerb in Gang und kleinere und mittlere Software-Anbieter in Deutschland bekommen eine echte Chance, am Software-Markt teilzunehmen.

Aus informatisch-technischer Perspektive lässt sich festhalten:

"Open Source"-Entwicklung ist bei Software eher − als proprietäre Software-Entwicklung − geeignet, mit zwei Grundproblemen moderner Software umzugehen:

1. zunehmende Komplexität
2. Fehlersuche und -Beseitigung

Die Entwicklung von Software im Unternehmen muss sich grundsätzlich nach betriebswirtschaftlichen Gesichtspunkten richten. Software wird aber in allen Bereichen der Gesellschaft zunehmend eingesetzt. Dadurch wirken sich betriebswirtschaftlich motivierte Entscheidungen zur Qualitätssicherung unmittelbar auf die Gesellschaft aus. Fehlentscheidungen führen dabei zu gravierenden Problemen, wenn es keine Ausweichmöglichkeit gibt.

> **Kein anderes Software-Entwicklungsverfahren hat in so kurzer Zeit so viel hochwertige Software hervorgebracht wie die "Open Source"-Community. Kein anderes Software-Entwicklungsverfahren kann so kurze Reaktionszeiten bei der Beseitigung sicherheitsrelevanter Fehler nachweisen. Und: Noch nie war Software so preiswert zu erstellen und zu verteilen.**

Mit der "Open Source"-Software-Entwicklung hat der Dialog Einzug gehalten in die lange Zeit exklusive Welt der Software-Entwicklung. Die Anwender haben sich Gehör und Einfluß verschafft. Das Verhältnis von Entwicklern und Anwender hat sich dadurch entschieden verändert, beide Gruppen haben davon − in unterschiedlicher Form − profitiert. Und: Die Qualität der Software hat sich verbessert.

5. Probleme bei der Durchsetzung von Patenten auf Software−Erfindungen

In diesem Kapitel werden einige Aspekte typischer Patentverletzungsszenarien im Zusammenhang mit software-bezogenen Erfindungen beleuchtet. Dabei wird grundsätzlich – soweit nicht anders angegeben – von der Geltung deutschen Rechtes ausgegangen. Bei einer angemessenen Würdigung ausländischen Rechtes in Fallkonstellationen mit Auslandsberührung wäre in Einzelfällen auch mit geringfügig abweichenden Ergebnissen zu rechnen. Dieses Kurzgutachtens behandelt diese Fälle nur exemplarisch.

5.1. Arten der Patentverletzung

Man unterscheidet *mittelbare* und *unmittelbare* Patentverletzungen.

Unmittelbare Patentverletzungen

Bei der *unmittelbaren Patentverletzung* sind alle Merkmale des verletzten Patentanspruches dem Wortsinne oder ihrer wesentlichen Wirkung nach in dem spezifischen Verletzungsgegenstand verwirklicht. Eine Einheit, bestehend aus Hardware und Software, die ein bestimmtes Verhalten zeigt, vollendet daher den Tatbestand der unmittelbaren Patentverletzung.

Nach deutschem Patentrecht monopolisiert ein Patent die Benutzung einer Erfindung in folgender Weise (§9 PatG):

> **«§9 [Wirkung des Patentes]**
>
> Das Patent hat die Wirkung, daß allein der Patentinhaber befugt ist, die patentierte Erfindung zu benutzen. Jedem Dritten ist es verboten, ohne seine Zustimmung
>
> 1.) ein Erzeugnis, das Gegenstand des Patents ist, herzustellen, anzubieten, in Verkehr zu bringen oder zu gebrauchen oder zu den genannten Zwecken entweder einzuführen oder zu besitzen.

2.) ein Verfahren, das Gegenstand des Patents ist, anzuwenden oder, wenn der Dritte weiß oder es auf Grund der Umstände offensichtlich ist, daß die Anwendung des Verfahrens ohne Zustimmung des Patentinhabers verboten ist, zur Anwendung im Geltungsbereich dieses Gesetzes anzubieten;

3.) das durch ein Verfahren, das Gegenstand des Patents ist, unmittelbar hergestellte Erzeugnis anzubieten, in Verkehr zu bringen oder zu gebrauchen oder zu den genannten Zwecken entweder einzuführen oder zu besitzen.»

Beispiel: Kraftfahrzeugbremsensteuerung I

Gegeben sei ein mit Wirkung für Deutschland erteilter Patentanspruch auf eine Vorrichtung, beispielsweise eine Kraftfahrzeugbremsensteuerung, oder auf ein Verfahren, beispielsweise ein Verfahren zur Steuerung eines Aufzuges. Weiterhin sei angenommen, die Merkmale der beanspruchten Kraftfahrzeugbremsensteuerung beziehungsweise des Aufzugssteuerungsverfahrens seien geeignet, um mittels eines Universalrechners (Mikroprozessor) technisch realisiert zu werden. Dann begeht jeder eine Patentverletzung, der eine fertige Baugruppe mit einem Mikroprozessor und einem Datenspeicher, in dem ein Datenverarbeitungsprogramm abgespeichert ist, das alle Merkmale des jeweiligen Patentanspruches erfüllt,

- in Deutschland *herstellt*
- in Deutschland *anbietet*
- in Deutschland *in Verkehr bringt*
- in Deutschland *gebraucht*
- *zwecks Herstellung, Anbietens, Inverkehrbringens oder Gebrauchens entweder einführt oder besitzt*.

Eingebettete Systeme

Dort, wo Datenverarbeitungsprogramme vor allem innerhalb von eingebetteten Systemen zum Einsatz kommen, wird der Programmierer traditionell ein Bewusstsein dafür besitzen, dass sich das später zu vermarktende Gesamtprodukt an den gewerblichen Schutzrechten der Wettbewerber wird messen lassen müssen. Üblicherweise wird daher bei der Erstellung der entsprechenden Steuerungssoftware bereits in der Konzeptionsphase dafür gesorgt, dass bekannte Schutzrechte Dritter nach Möglichkeit umgangen werden. Insoweit bereitet die Existenz von Patenten auf software-bezogene Erfindungen diesen Wirtschaftskreisen keine prinzipiellen Probleme.

In anderen Bereichen – wie beispielsweise dem "Open Source"-Betriebssystem Linux – sind Datenverarbeitungsprogramme jedoch als vermarktungsfähige Wirtschaftsgüter anzusehen.

Auch die Datenverarbeitungsprogramme innerhalb von eingebetteten Systemen könnten theoretisch bei Bedarf eigenständig vermarktet werden.

Mittelbare Patentverletzungen

Ein Patentverletzer könnte den Patentanspruch umgehen wollen, indem er einen patentverletzenden Gegenstand in zwei oder mehr Teilsysteme aufteilt, die – jedes für sich genommen – keine Verletzung darstellen. Der Endanwender würde dann angeleitet, die für sich genommen nicht patentverletzenden Teilsysteme zu dem patentverletzenden Gesamtsystem zusammenzusetzen. Insbesondere wenn der Endanwender das wieder zusammengesetzte Gesamtsystem typischerweise im privaten Bereich zu nichtgewerblichen Zwecken benutzt − und daher Kraft der Schrankenbestimmungen des Patentrechtes nicht von den Wirkungen des Monopolrechtes berührt wird −, liefen die Ansprüche des Patentinhabers de facto ins Leere, da bei einem derartigen Szenario die gewerblichen Hersteller und Vertriebsorganisationen nur Fragmente herstellen beziehungsweise vermarkten, die für sich genommen nicht alle Merkmale des Patentanspruches verwirklichen. Bei der mittelbaren Patentverletzung wird nun darauf abgestellt, dass auch ein Teilsystem durchaus eine Verletzungsform darstellen kann, wenn es subjektiv bestimmt und objektiv geeignet ist, die im Patentanspruch definierte geschützte Erfindung zu verwirklichen, vorausgesetzt, bei dem Teilsystem handelt es sich in diesem Zusammenhang um ein "wesentliches Mittel".

Der Gesetzgeber hat diese Möglichkeit als Umgehungstatbestand erkannt und dagegen den eigenständigen Patentverletzungstatbestand der mittelbaren Patentverletzung im Patentgesetz verankert:

> **«§10 [Verbotene Verwendung von Mitteln zur Benutzung der Erfindung]**
>
> (1) Das Patent hat ferner die Wirkung, daß es jedem Dritten verboten ist, ohne Zustimmung des Patentinhabers im Geltungsbereich dieses Gesetzes anderen als zur Benutzung der patentierten Erfindung berechtigten Personen Mittel, die sich auf ein wesentliches Element der Erfindung beziehen, zur Benutzung der Erfindung im Geltungsbereich dieses Gesetzes anzubieten oder zu liefern, wenn der Dritte weiß oder es auf Grund der Umstände offensichtlich ist, daß diese Mittel dazu geeignet und bestimmt sind, für die Benutzung der Erfindung verwendet zu werden.

(2) Absatz 1 ist nicht anzuwenden, wenn es sich bei den Mitteln um allgemein im Handel erhältliche Erzeugnisse handelt, es sei denn, daß der Dritte den Belieferten bewußt veranlaßt, in einer nach §9 Satz 2 verbotenen Weise zu handeln.

(3) Personen, die in §11 Nr. 1 bis 3 genannten Handlungen vornehmen, gelten im Sinne des Absatzes 1 nicht als Personen, die zur Benutzung der Erfindung berechtigt sind.»

Die geräteunabhängige Vermarktung von Datenverarbeitungsprogrammen schiene somit – zumindest auf den ersten Blick – ohne den Verletzungstatbestand der mittelbaren Patentverletzung geeignet, die Möglichkeit einer *Patentumgehung* zu eröffnen.

Beispiel: Kraftfahrzeugbremsensteuerung II

Angenommen, eine Kraftfahrzeugbremsensteuerung wird außerhalb Deutschlands in einem Land hergestellt, in dem es keinen Patentschutz für die hier betrachtete Erfindung gibt. Dann läge es nahe, die Steuerungsbaugruppe zunächst ohne den Speicherbaustein zu fertigen und nach Deutschland zu importieren. Die zollamtliche Abfertigung und der Vertrieb einer Vorrichtung, die lediglich einen allseits anwendbaren Mikroprozessor sowie andere (hier patentrechtlich nicht relevante) Standardbausteine umfasst, bereitet keinerlei Patentverletzungsrisiken.

Die hiervon strikt getrennte Einfuhr und Vermarktung des Speicherbausteins mit dem Datenverarbeitungsprogramm, durch das der Steuerungsvorrichtung überhaupt erst patentverletzende Eigenschaften verliehen werden, verletzt als solches gewiss auch nicht den Patentanspruch.

Die patentverletzende Funktionalität wird erst sich im Zusammenwirken des Speicherbausteins mit der Hardware der Steuerungsvorrichtung verwirklicht. Der Importeur könnte zum Beispiel dem im Inland vermarkteten Speicherbaustein einen "Beipackzettel" beifügen, mit dem der Endkunde angewiesen wird, durch geeignetes Einfügen des Speichers in die Hardware der Steuerungsvorrichtung die patentverletzende Funktionalität herzustellen.

Dies hätte für Hersteller und Importeur wesentliche wirtschaftliche Vorteile: Die Patentverletzung geschähe erst beim *Endanwender* durch die Endmontage des Speicherbausteines. Der rein private Gebrauch beim nichtkommerziellen Letztverbraucher zu nichtgewerblichen Zwecken wäre infolge der Schrankenbestimmungen des Patentrechtes nicht als deliktische Patentverletzung anzusehen. Auch wäre die übergroße Vielzahl der Patentverletzungen bei kommerziellen Anwendern – beispielsweise bei einem geschäftlich benutzten Kraftfahrzeug – durch den Patentinhaber zivilrechtlich praktisch kaum flächendeckend zu ahnden. Im Ergebnis wäre der Patentschutz aus der Sicht des Patentinhabers entscheidend geschwächt oder sogar völlig entwertet.

Die Vermarktung des Speicherbausteines im Inland zusammen mit einem "Beipackzettel" (s. obiges Beispiel) stellt aufgrund der Sonderbestimmung in §10 PatG eine klare Patentverletzung dar, denn:

- Das im Speicherbaustein abgespeicherte Datenverarbeitungsprogramm stellt ein "*wesentliches Mittel*" der Erfindung dar – die den neuartigen technischen Effekt der Bremsensteuerung beschreibenden Merkmale des Patentanspruches werden gerade genau dann

verletzt, wenn das Datenverarbeitungsprogramm mit der Hardware der Steuerungsvorrichtung (präziser: mit dem darin enthaltenen Mikroprozessor) zusammenwirkt.

- Der Speicherbaustein ist – wie man bei Bedarf durch Erproben einfach feststellen kann – auch ohne weiteres technisch *geeignet*, der Hardware der Steuerungseinrichtung die patentverletzenden Merkmale zu verleihen.
- Bei der vorliegenden Fallkonstellation macht der mitgelieferte "Beipackzettel" unzweifelhaft klar, dass der Hersteller und/oder der Importeur den Speicherbaustein zur Vollendung der patentverletzenden Steuerung *bestimmt* hatten, d.h. dass hier nicht bloß Endabnehmer aus eigener Kreativität heraus eine − vom Hersteller bzw. Importeur aufgrund einer zuvor nicht ins Auge gefassten objektiven Eignung desselben nicht vorherzusehende und daher auch nicht zu verantwortende − "Zweckentfremdung" des Speicherbausteins betreiben.

Im Ergebnis wird durch die einschlägige Wirtschaft daher bei Datenverarbeitungsprogrammen für eingebettete Systeme keine patentrechtlich motivierte eigenständige Software-Vermarktung stattfinden, da der Tatbestand der *mittelbaren Patentverletzung* hier greift und jeglichen Vorteil aus einer entsprechenden Patentumgehung vereitelt.

Download von Software

Bei Software, die typischerweise als eigenständiges Wirtschaftsgut vertrieben wird, wächst gegenwärtig die Bedeutung des Herunterladens der Programmdateien von einem Serverrechner über das Internet auf den Rechner des Anwenders.

In allen Fällen, in denen ein Datenverarbeitungsprogramm als wesentliches Mittel anzusehen ist, begeht eine mittelbare Patentverletzung einer patentgeschützen Erfindung, wer

- das Programm zum Herunterladen anbietet oder durch den Vollzug des Vorganges des Herunterladens vom eigenen Server liefert und
- weiß oder auf Grund offensichtlicher Umstände wissen müsste, dass dieses Programm dazu geeignet und bestimmt ist, für die Benutzung der Erfindung verwendet zu werden.

Der Begriff des Anbietens schließt das Feilbieten zum Erwerb oder zur Benutzung ein. Das Anbieten ist eine Handlung, die Dritte anregen soll, ein Erzeugnis oder ein Verfahren zum Eigentum oder zur Benutzung zu erwerben.[179] Die Kenntnis des Patentschutzes der Erfindung durch den Verletzer ist selbstverständlich nicht erforderlich.

5.2. Die Schranken des Patentrechtes

Die Unentgeltlichkeit beim Umgang mit "Freier Software" im Internet konstituiert nicht automatisch den Patentschutz durchbrechende Ausnahmetatbestände infolge *rein im privaten Bereich stattfindender*, *nicht zu gewerblichen Zwecken erfolgender* Patentbenutzung etwa gemäß §11 Nr. 1 PatG −

179 Vgl. **Benkard 1993** zu §9, Rdn. 42, S. 422.

bei einer auf Dauerhaftigkeit angelegten Software-Distribution über das Internet dürfte regelmäßig kein privates Handeln vorliegen.[180]

Das durch das Patent geschaffene Ausschließlichkeitsrecht ist nicht schrankenlos. Ein wichtiger Katalog von Schrankenbestimmungen findet sich in §11 PatG:

«**§11 [Erlaubte Handlungen]**

Die Wirkung des Patents erstreckt sich nicht auf

1. Handlungen, die im privaten Bereich zu nichtgewerblichen Zwecken vorgenommen werden;
2. Handlungen zu Versuchszwecken, die sich auf den Gegenstand der patentierten Erfindung beziehen;
3. die unmittelbare Einzelzubereitung von Arzneimitteln in Apotheken auf Grund ärztlicher Verordnung sowie auf Handlungen, welche die auf diese Weise zubereiteten Arzneimittel betreffen;
4. den an Bord von Schiffen eines anderen Mitgliedstaates der Pariser Verbandsübereinkunft zum Schutz des gewerblichen Eigentums stattfindenden Gebrauch des Gegenstands der patentierten Erfindung im Schiffskörper, in den Maschinen, im Takelwerk, an den Geräten und sonstigem Zubehör, wenn die Schiffe vorübergehend oder zufällig in die Gewässer gelangen, auf die sich der Geltungsbereich dieses Gesetzes erstreckt, vorausgesetzt, daß dieser Gegenstand dort ausschließlich für die Bedürfnisse des Schiffes verwendet wird;

180 Vgl. **Benkard 1993** zu §11, Rdn. 3ff, S. 455: «Regelmäßig ist jedoch [...] auch beim Anbieten und Liefern von Mitteln, z.B. wesentlichen Elementen geschützter Erfindungen, zur Benutzung der Erfindung (§10 PatG), der private Bereich verlassen, weil mit ihnen die Befriedigung fremder Bedürfnisse bezweckt wird (§10 Abs. (3) PatG). Der private Bereich ist nicht nur dann verlassen, wenn eine Handlung im gewerblichen Bereich vorgenommen wird, sondern auch dann, wenn öffentliche Institutionen die Handlung vornehmen, ohne damit einen gewerblichen Zweck zu verfolgen [...] Handlungen zu gewerblichen Zwecken werden von der Wirkung des Patentes erfaßt, gleichgültig ob sie in oder außerhalb des privaten Bereiches vorgenommen werden.»

5. den Gebrauch des Gegenstandes der patentierten Erfindung in der Bauausführung oder für den Betrieb der Luft- oder Landfahrzeuge eines anderen Mitgliedstaates der Pariser Verbandsübereinkunft zum Schutz des gewerblichen Eigentums oder des Zubehörs solcher Fahrzeuge, wenn diese vorübergehend oder zufällig in den Geltungsbereich dieses Gesetzes gelangen;

6. die in Artikel 27 des Abkommens vom 7. Dezember 1944 über die internationale Zivilluftfahrt (BGBl. 1956 II S. 411) vorgesehenen Handlungen, wenn diese Handlungen ein Luftfahrzeug eines anderen Staats betreffen, auf den dieser Artikel anzuwenden ist.»

Selbstverständlich müssen sich auch solche Unternehmen, die ihre Software-Produkte nach dem "Closed Source"-Verfahren durch entgeltliche Lizenzierung vermarkten, auf die Gegebenheiten des Patentsystems einstellen. Die mit der entgeltlichen Lizenzierung von Software generierten Geldflüsse gestatten diesen Unternehmen jedoch in aller Regel eine mehr oder minder aktive Patentrechtspolitik durch Recherchen nach dem Stand der Technik und Schutzrechten Dritter sowie durch Verfolgung eigener Patentanmeldungen und Verteidigung eigener Rechtspositionen. Ein System der regelmäßigen Gewährung von Patenten auf Software-Erfindungen führt somit zu einer *konzeptionellen Asymmetrie* gegenüber den unterschiedlichen Software-Entwicklungs- und -Vermarktungsmodellen.

Zahllose Programmierer lassen ihre Algorithmen in den Quelltext einfließen. Es ist deshalb praktisch aussichtslos, die Tätigkeit und das Ergebnis zu kontrollieren. Man muss wohl davon ausgehen, dass auch die Firmen durch "Software-Patente" behindert werden, die wie Red Hat oder SuSE durch die Vermarktung von Software-Distributionen beträchtliche Geldflüsse generieren.

5.3. Unterlassungs- und Schadensersatzansprüche gegen Akteure im Umkreis der "Open Source"−Software

Unterlassungsansprüche

Der bereits erläuterte §10 PatG bildet in Verbindung mit §139 Absatz 1 PatG die Grundlage für Unterlassungsansprüche von Patentinhabern gegen jeden, der über das Internet Datenverarbeitungsprogramme zum Herunterladen *anbietet*, wenn die näheren Umstände gemäß den Tatbestandsmerkmalen dieser Bestimmung erfüllt sind.

Dies bedeutet zunächst, dass der Patentinhaber verlangen kann, dass die Datei, die das mittelbar patentverletzende Datenverarbeitungsprogramm implementiert, vom Server heruntergenommen und somit nicht weiter zum Herunterladen angeboten wird. Dieser Unterlassungsanspruch hat als solcher auch für finanziell minderprivilegierte Akteure aus Kreisen der nichtkommerziellen "Open Source"-Entwicklergemeinschaft zunächst noch keine nachteiligen finanziellen Folgen. Es ist jedoch damit zu rechnen, dass Patentinhaber ihre Ansprüche zunächst auf dem Wege der *Abmahnung* geltend machen und die pro Abmahnungsfall unmittelbar entstehenden Kosten in einer Größenordnung von bis zu mehreren Tausend Deutschen Mark auf die Patentverletzer abwälzen werden.

Die Rechtsfolgen einer Patentverletzung, nämlich der Unterlassungsanspruch und der Schadensersatzanspruch, sind im Patentgesetz in §139 PatG geregelt:

> «**§139 [Unterlassungs- und Schadensersatzanspruch]**
>
> (1) Wer entgegen den §§9 bis 13 eine patentierte Erfindung benutzt, kann vom Verletzten auf Unterlassung in Anspruch genommen werden.
>
> (2) Wer die Handlung vorsätzlich oder fahrlässig vornimmt, ist dem Verletzten zum Ersatz des daraus entstandenen Schadens verpflichtet. Fällt dem Verletzer nur leichte Fahrlässigkeit zur Last, so kann das Gericht statt des Schadensersatzes eine Entschädigung festsetzen, die in den Grenzen zwischen dem Schaden des Verletzten und dem Vorteil bleibt, der dem Verletzer erwachsen ist.
>
> (3) Ist Gegenstand des Patents ein Verfahren zur Herstellung eines neuen Erzeugnisses, so gilt bis zum Beweis des Gegenteils das gleiche Erzeugnis, das von einem anderen hergestellt worden ist, als nach dem patentierten Verfahren hergestellt. Bei der Erhebung des Beweises des Gegenteils sind die berechtigten Interessen des Beklagten an der Wahrung seiner Herstellungs- und Betriebsgeheimnisse zu berücksichtigen.»

Schadensersatzansprüche

Gravierender könnte im Einzelfall der aus §139 Absatz 2 PatG resultierende Anspruch des Patentinhabers auf *Schadensersatz* werden. Eine Pflicht zum Leisten von Schadensersatz entsteht nur bei *Verschulden*, also bei Vorsatz, oder bei *Fahrlässigkeit*, d.h. mangelnder Sorgfalt. Bei der Feststellung des Verschuldens kommt es immer auf die Verhältnisse des Einzelfalls an. Allgemein lässt sich jedoch sagen, dass die Kenntnis der für sein Fachgebiet einschlägigen Patente und Patentanmeldungen zumindest von jedem größeren Unternehmen erwartet wird. Fahrlässig ist es daher in

der Regel, wenn ein Fabrikant die Patentanmeldungen und -erteilungen in seinem Fachgebiet nicht verfolgt. Vom Benutzer eines patentverletzenden Gegenstandes wird nicht die gleiche strenge Sorgfalt gefordert wie vom Hersteller.[181]

Im Hinblick auf die verfügbare Rechtsprechung ist derzeit noch völlig offen, welche Sorgfaltsanforderungen an Entwickler oder Distributoren von "Open Source"-Software zu stellen sind, die Datenverarbeitungsprogramme unentgeltlich zum Herunterladen anbieten. Eine wichtige Rolle wird hier die eigene Sachkunde des Verletzers spielen. Es wäre nicht überraschend, wenn es insbesondere in Entwickler- oder Distributionsparks als fahrlässig anzusehen wäre, wenn eine Verletzung trotz der Kenntnis bekannter und öffentlich erörterter Patente geschieht, zumal wenn die Verletzung bereits bei Anwendung geringer Sorgfalt hätte vermieden werden können. Dies wäre dann der Fall, wenn die Verletzungsform wörtlich durch einen Patentanspruch erfasst würde, was u.U. auch ohne anwaltliche Beratung erkennbar sein kann.

5.4. Recherche nach Patentrechten Dritter und nach dem Stand der Technik

Das Funktionieren des Patentwesens ist davon abhängig, dass der für die Beurteilung der Patentfähigkeit nach den §§3 u. 4 PatG relevante Stand der Technik insbesondere im patentamtlichen Prüfungsverfahren tatsächlich mit ausreichender Genauigkeit objektiv ermittelbar ist.[182] Dem steht nicht prinzipiell entgegen, dass kein von einem Patentamt erstellter Recherchebericht lückenlos ist und dass möglicherweise sogar im Einspruchs- oder Patentnichtigkeitsverfahren ein an sich vorhandener Stand der Technik von Fall zu Fall aus subjektiver Unkenntnis nicht aufgedeckt wird. Die allgemein anzunehmende Verständlichkeit der für den einschlägigen Fachmann in Betracht kommenden Offenbarungsquellen eröffnet im Zusammenwirken mit dem Ordnungsinstrumentarium der Patentklassifikation sowohl dem Patentinhaber als auch der Öffentlichkeit eine faire Chance, im Einzelfall den relevanten Stand der Technik auch tatsächlich zu identifizieren.

Formalisierung bei biotechnologischen Sequenzen als Beispiel

Grundsätzliche Probleme beim Recherchieren nach dem Stand der Technik sind zuerst aus einem anderen patentrechtlichen Bereich, nämlich der Patentierung von biochemischen Sequenzen, offenbar geworden. Es wurde deutlich, dass auch der Fachmann ohne zusätzliche Hilfsmittel nicht mehr in der Lage sein würde, anhand einer vorgegebenen Sequenz alle relevanten bekannten identischen oder ähnlichen Sequenzen zu ermitteln. Für das manuelle Sichten langer Sequenzlisten ist der menschliche Geist offenbar nicht eingerichtet. Dieses Problem konnte – was die Biotechnologie anbetrifft – durch den Aufbau von Sequenzdatenbanken gelöst werden, denn die Universalität des genetischen Codes ermöglicht eine Recherche mittels einer standardisierten ("kanonisierten") Datenbankabfrage.

---

181 **Benkard 1993** zu §139, Rdn. 42ff.

182 Vgl. hierzu insbesondere **Horns 2001**.

Jeder Universalrechners hat die Eigenschaft, alle anderen Universalrechner durch eine geeignete Software nachbilden (emulieren) zu können. Es kann jedoch keinen "kanonischen" Code zur Niederschrift von Algorithmen geben, die − als linguistisches Konstrukt − in Programmiersprachen codiert werden. Neben den marktgängigen realen Hardwareprozessoren wären im Prinzip *alle denkbaren virtuellen Prozessoren* zu berücksichtigen, die ebenfalls zum Stand der Technik beitragen können.

Somit ist es nicht möglich, wie beispielsweise bei Gensequenzen, die Recherche nach dem Stand der Technik automatisiert durchzuführen. Dies könnte sich als ernsthaftes Hindernis für eine liberalere Gewährung von Patenten auf computer-implementierbare Erfindungen erweisen,[183] sofern der einschlägige Stand der Technik zu einem erheblichen Teil als linguistisches Programmcode--Konstrukt vorliegt. Sollten sich die derzeitigen Probleme als pragmatisch lösbar erweisen, etwa durch vermehrtes und besser geschultes Personal oder durch neuartige Recherchеverfahren,[184] könnte zumindest für die Zukunft eine höhere Rechtssicherheit erwartet werden.

Es scheint aber derzeit mangels überzeugender Vorschläge für konkrete Verbesserungsmaßnahmen nicht mehr ausgeschlossen, dass ein grundlegendes Problem vorliegt, welches aus prinzipiellen Gründen im Zusammenhang mit der Natur der Universalrechner nicht lösbar ist.[185]

5.5. Patentverletzung durch Distribution von "Open Spource"−Software

Da in der Praxis die Entwicklungskooperation und die Distribution von "Open Source"-Software häufig über das Internet abgewickelt wird, ist von Bedeutung, dass derjenige, der die über das Netz zur Verbreitung bereitgestellten Inhalte rechtlich zu verantworten hat, *potentiell* mit patentrechtlichen Ansprüchen aus allen über das Internet erreichbaren Jurisdiktionen rechnen muss.[186]

In jedem Land, in dem in das Netz gestellte Inhalte abrufbar sind, kann derjenige, der diese Inhalte rechtlich zu vertreten hat, gerichtspflichtig werden. Im Zeitalter des Internet ist besonders

---

183 Vgl. dazu u.a. **van Raden 1993**, S. 454, m.w.N.

184 Etwa durch eine zweckdienliche Umgestaltung der Internationalen Patentklassifikation (IPC).

185 Allenfalls wäre etwa zu erwägen, für Patente auf Software-Erfindungen nach dem Vorbild der Gentechnologieerfindungen zu verlangen, dass die erfindungsgemäße Lösung in der Patentanmeldung in einer bestimmten, gesetzlich vorschriebenen und somit kanonisierten Form eingereicht wird. Ein möglicher Kandidat für eine gesetzlich kanonisierte Darstellungsform ist die Unified Modelling Language (UML). S. dazu im Internet: http://www.rational.com/; http://www.omg.org/cgi-bin/doc?ad/99-06-08.pdf (1.11.2000).

186 Vgl. **Esslinger/Betten 2000**. Die Autoren weisen zutreffend darauf hin, dass nach geltender Rechtsprechung bei Angeboten im Internet für den Ort der deliktischen Handlung nicht auf den Ort abzustellen ist, an dem die Einrichtung der Homepage erfolgt ist oder an dem der Server steht. Als Begehungsort kommt grundsätzlich jeder Ort in Betracht, von dem aus die Homepage bestimmungsgemäß abgerufen und einen Rechtsverstoß bewirkt werden kann. Sie verweisen auf das Vorbenutzungsrecht gemäß §12 PatG, der denjenigen *im Inland* von der Wirkung eines Patentes ausnimmt, der zur Zeit der Anmeldung bereits *im Inland* die Erfindung in Benutzung genommen oder die dazu erforderlichen Veranstaltungen getroffen hatte. Diese Begünstigung ist jedoch zweifach territorial begrenzt: Zum einen tritt die Begünstigung nicht im Ausland ein, und zum anderen kann sie nur durch Handlungen im Inland erworben werden. In Anbetracht der Globalität des Internet dürfte diese Bestimmung für die Software-Distribution nur eine geringe Bedeutung erlangen.

problematisch, dass die Regeln, nach denen die Gerichtspflichtigkeit in einem bestimmten Land festgestellt wird, nicht international harmonisiert sind. In jedem Land, in dem Inhalte über das Internet abgerufen werden können, entscheiden dortige Gerichte nach lokal geltendem Verfahrensrecht, ob sie sich für zuständig halten oder nicht.

Beispiel: Gerichtspflichtigkeit

Angenommen, in Deutschland existiert ein bestimmtes Patent, das auf dem Territorium der Bundesrepublik Deutschland mittelbar durch ein bestimmtes Datenverarbeitungsprogramm verletzt werden kann. Die in Deutschland patentrechtlich geschützte Erfindung sei in den Vereinigten Staaten patentfrei. Wenn in den Vereinigten Staaten ein Server betrieben wird, auf dem das Datenverarbeitungsprogramm zum Abruf über das Internet bereitgehalten wird, kann der "Primärstörer", dem die Verantwortung für das Anbieten des Datenverarbeitungsprogrammes zum Herunterladen über das Internet rechtlich zuzurechnen ist, vom Patentinhaber *vor einem deutschen Gericht* auf Unterlassung und Schadensersatz in Anspruch genommen werden.

Das Gericht wird dann nach deutschem "Internationalem Privatrecht" (IPR) prüfen, ob es sich für zuständig hält. Wenn der Patentinhaber beispielsweise vorträgt, das Datenverarbeitungsprogramm sei in Deutschland rechtswidrig heruntergeladen worden, könnte das Gericht – unter bestimmten näheren Umständen – einen Gerichtsstand am deutschen Deliktsort (deliktischer Gerichtsstand) annehmen und sich für zuständig erklären. Der verantwortliche Betreiber des Servers hätte sich dann vor einem deutschen Gericht zu verantworten, unabhängig davon, wo er seinen Sitz oder Wohnort hat. Obsiegt der Patentinhaber, stünde er freilich vor dem Problem der effektiven Vollstreckung des erstrittenen Titels: Nur wenn entsprechende zwischenstaatliche Abkommen zur gegenseitigen Anerkennung von Gerichtsentscheidungen bestehen, wird sich das Urteil auch vollstrecken lassen. Umgekehrt besteht für Software-Entwickler und -Distributoren, die in Deutschland ihren Sitz oder Wohnort haben, ein entsprechendes Risiko, irgendwo im Ausland wegen Patentverletzung belangt zu werden, wenn sich ein dortiges Gericht für zuständig erklärt.

Dieses Risiko ließe sich nur dann völlig vermeiden, wenn vor der Bereitstellung eines Datenverarbeitungsprogrammes im Internet durch umfangreiche Patentrecherchen sichergestellt würde, dass durch das Anbieten der Software zum Herunterladen in keiner Jurisdiktion der Erde ein Rechtsverstoß begangen wird. Praktisch ist dies selbst für Großunternehmen, geschweige denn für nichtkommerziell agierende "Open Source"-Aktivisten, völlig unmöglich. Man würde die Anforderungen an die verkehrsübliche Sorgfalt übertreiben, wenn man jedem Internet-Inhalteanbieter derartige Pflichten auferlegen würde.

Im Ergebnis bedeutet dies jedoch, dass jeder, der ein Datenverarbeitungsprogramm im Internet zum Herunterladen anbietet, ein schwer kalkulierbares Restrisiko eingeht. Schlimmstenfalls findet man sich in einem weit entfernten Ausland plötzlich in der Beklagtenrolle wieder und steht vor der Wahl, entweder mit hohem finanziellen Aufwand über einen lokalen Rechtsanwalt die eigenen Interessen wahrnehmen zu lassen oder die völlig unkalkulierbaren Folgen eines in Abwesenheit des Beklagten ergehenden Versäumnisurteils hinnehmen zu müssen.

Diese Problematik betrifft selbstverständlich nicht nur "Open Source"-Software. Der offengelegte Quelltext zeigt jedoch ohne weiteres die Funktionalität der Software. Das erleichtert dem Patentinhaber das Beschreiten des Klageweges erheblich. Eine kompromisslose Durchsetzung aller Rechtsansprüche aus allen Jurisdiktionen gegenüber allen über das Internet verbreiteten Inhalten würde die Idee des Internet an sich in Frage stellen.

Glücklicherweise gibt es auch seitens der Rechtsprechung Ansätze, die Gerichtspflichtigkeit im Ausland zu begrenzen. Ein wichtiger Aspekt ist dabei die *objektiv erkennbare Absicht* des Inhalteanbieters, einen Inhalt nur an Nutzer in bestimmten Ländern zu adressieren.

Wer als Anbieter nur solche Kunden zulässt − beispielsweise nach einer Identifikation − die ihren Sitz oder Wohnort in Ländern mit unproblematischer Rechtsprechung haben, hat unter praktischen Gesichtspunkten maximale Sorgfalt walten lassen. Er sollte dann auf eine Rechtsanwendung hoffen dürfen, bei der er aufgrund einer Selbstbeschränkung der Gerichte nicht haftbar gemacht wird.

Doch der "Open Source"-Community wird dieser Aspekt in der Praxis kaum helfen. Es erscheint schwer vorstellbar, dass ein bestimmtes Mitglied der Entwicklergemeinschaft von der weiteren Mitarbeit an einem bestimmten "Open Source"-Projekt ausgeschlossen wird, weil er seinen Wohnort in einem Land hat, in dem das in der Entwicklung befindliche Datenverarbeitungsprogramm für patentverletzend gehalten wird.

5.6. Die Durchsetzung von Patentrechten und das Grundrecht auf Kommunikationsfreiheit

Bis Anfang des Jahres 2000 war in den Vereinigten Staaten eine Debatte um einen eigentümlichen Tatbestand festzustellen. Der Streit hat zwar keinen unmittelbaren Bezug zu Fragen des gewerblichen Rechtsschutzes, er könnte aber dennoch Auswirkungen auf die Distribution von Software über das Internet haben.

Die U.S.-Verfassung beinhaltet einen sehr weitgespannten Schutz der "Freien Rede" ("Free Speech"):

> «**Amendment I to the Constitution of the United States of America**
>
> Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof, or abridging the free speech, or of the press, or the right of the people peacefully to assemble, and to petition for a redress of grievances.»

Dieser Schutz umfasst beispielsweise auch das Recht, Bücher mit Beschreibungen kryptographischer Verfahren sowie entsprechende Programmlistings *als Druckwerke* nicht nur im U.S.-Inland zu vertreiben, sondern auch *zu exportieren*.

Kryptographische Vorrichtungen werden gemäß U.S.-Recht jedoch grundsätzlich als (kriegs-) waffenähnliche Gegenstände eingestuft und dürfen nur gemäß den strengen Ausfuhrbestimmungen für Kriegswaffen exportiert werden. Kryptographische Datenverarbeitungsprogramme, sowohl im Quelltext als auch in kompilierter Form, sind kryptographischen Vorrichtungen gleichgestellt, da man mit ihnen ebenfalls verschlüsseln kann. Dieser – vom Patentrecht völlig unabhängige – Zusammenhang führte u.a. zu der bemerkenswerten Situation, dass zwar ein Standardlehrbuch über Kryptographie mit zahlreichen Programmlistings als Druckwerk auch in das Ausland exportiert werden durfte,

die dazugehörige Computerdiskette mit den Programmen in maschinenlesbarer Form durfte wegen der Exportbestimmungen jedoch nicht ausgeführt werden. Auch war die Verbreitung der Programmlistings über das Internet nach außerhalb der Vereinigten Staaten strafbar.[187]

In einer kürzlich ergangenen Entscheidung[188] hat ein amerikanisches Revisionsgericht festgestellt, dass Mitteilungen in Gestalt von Software unter das First Amendment fallen, d.h. *den Schutz für die "Freie Rede"* genießen:

> «[...] Likewise, computer source code, though unintelligible to many, is the preferred method of communication among computer programers. Because computer source code is an expressive means for the exchange of information and ideas about computer programming, we hold that it is protected by the First Amendment.»

Im Hinblick auf die Gewährung von Patenten auf software-bezogene Erfindungen, mit denen der Patentinhaber insbesondere auch gegen die Verbreitung von (mittelbar) patentverletzender Software durch Dritte − beispielsweise über das Internet − vorgehen könnte, könnten sich Patentinhaber veranlasst sehen, die Entfernung von "Source Code" von Websites zu fordern, wenn dieser unter einen erteilten Patentanspruch zu fallen scheint.

Eine rein patentrechtliche Betrachtung könnte jedoch leicht übersehen, dass der Software-Quelltext in der Praxis stets in mehr oder minder ausgeprägtem Maße eine Doppelfunktion besitzt:

- Zum einen ist er ein an einen Prozessor gerichtetes Steuerungsagens.
- Zum anderen dient er – in der Regel mit Kommentaren in einer natürlichen Sprache durchsetzt − als Kommunikationsmittel zwischen Software-Entwicklern.

Rechtlich schwierig wird es, wenn jemand Eigenschaften von Algorithmen wissenschaftlich erörtern möchte und hierzu die präzise Darstellungsform eines Quelltextes wählt. Die Ausführung des aus dem Quelltext compilierten Programms auf einem Computer würde eine Patentverletzung darstellen.

---

187 Inzwischen sind die U.S.-Exportbestimmungen gelockert worden, so dass bestimmte Kryptographieprogramme für den zivilen Alltagsgebrauch (z.B. PGP) über das Internet exportiert werden dürfen. Die nach der politischen Liberalisierung der Exportvorschriften ergangene Gerichtsentscheidung *in re Junger* ist dennoch von großem Interesse, da sie die Eigenschaft von im Quelltext vorliegenden Datenverarbeitungsprogrammen beleuchtet, *Ausdrucksmittel für zwischenmenschliche Kommunikation* sein zu können.

188 **Junger v. Daley 2000**.

Die Kommunikation zwischen Software-Entwicklern erfordert aber in der Praxis das Gespräch über den Quelltext. Es könnte also zu einer Kollision zwischen dem Unterlassungsanspruch des Patentinhabers und den Informationsrechten des Art. 5 des Grundgesetzes kommen.

Es ist ungeklärt und bedarf dringend der wissenschaftlichen Untersuchung, wie dieser Konflikt aufgelöst werden kann.

6. Das "Open Source"−Paradigma und die Sicherheit in der Informationstechnik

Sicherheit in der Informationsgesellschaft ist ohne sichere Software nicht erreichbar. Doch wie kann Sicherheit der Software erreicht werden? Kann "Open Source"-Software einen Beitrag dazu leisten? Und wenn ja, welchen? Dieses Kapitel untersucht die Probleme und mögliche Wege zu ihrer Lösung.

6.1. Funktionalität versus Sicherheit − Das Testproblem

> «Normal security testing fails for several reasons. First, security flaws can appear anywhere. They can be in the trust model, the system design, the algorithms and protocols, the implementation, the source code, the human−computer interface, the procedures, or the underlying computer system (hardware, operating system, or other software. ..., these flaws cannot be found through normal beta testing. Security has nothing to do with functionality.»[189]

> «Security testing is quite different from typical software testing. Normal software testing checks for the accuracy of output from given input and tests for the reporting of specific errors. However, security testing deals with how the system responds to the unexpected.»[190]

Die Komplexität der Software ist der ausschlaggebende Faktor für deren Verlässlichkeit und Sicherheit geworden.[191]

Moderne Software ist hoch komplex und verfügt über eine Vielzahl von Funktionen. Diese Funktionen interagieren zum Teil miteinander, zum Teil mit dem Anwender. Die Interaktion erfolgt bei korrekter Implementierung und korrektem Anwenderverhalten getreu der Spezifikation.

Die Tests, die durchgeführt werden, um festzustellen, ob die Spezifikation eingehalten wird, setzen die Kenntnis eben dieser Spezifikation voraus. Unter der Annahme, dass alle von der Spezifikation vorgesehenen Fälle erfolgreich getestet wurden, kann eine Software als fehlerfrei gelten − aber nur unter den spezifizierten Bedingungen. Funktionale Tests liefern daher ausschließlich Aussagen zur Funktionsqualität, nicht zum Sicherheitsniveau.

---

189 **Schneier 2000**, p. 334f.

190 **Pipkin 2000**, p. 41.

191 Vgl. **Schneier 2000**, p. 354ff; **Pipkin 2000**, p. 40.

Hinzu kommt: Die Vorgaben in der Spezifikation stammen bei proprietärer Standardsoftware vom Hersteller und nicht von den Anwendern. Ob die Anwendererwartungen bezüglich des Software-Verhaltens erfüllt werden, findet dabei nur bedingt Berücksichtigung. Insofern kann auch das Anwenderverhalten in den Tests nur partiell simuliert werden.

Solange es keine eigene Sicherheitsspezifikation gibt, wird es auch keine Sicherheitsimplementierung und keine aussagekräftigen Tests zur Sicherheit einer Software geben. Sicherheitsspezifikationen gehören aber gegenwärtig nicht zum "Handwerkszeug" der Software-Entwickler. Dieser Mangel ist sicherlich auch auf das Fehlen einheitlicher, verbindlicher Sicherheitsstandards für Software zurückzuführen.

Die Betrachtung von Software aus der Perspektive der IT-Sicherheit deckt also ein Dilemma auf:

> *Ohne Kenntnis der Spezifikation kann man eine Software nicht auf Sicherheit testen. Die Kenntnis dieser Spezifiktion, gar deren Veränderung, ist aber bei proprietärer Software gerade ausgeschlossen.*

Eine im Wachsen begriffene Anzahl von Fachleuten auf dem Gebiet der IT-Sicherheit ist der Auffassung, dass dieser Konflikt nur durch die Benutzung quellenoffener Software aufgelöst werden kann. Das ist auch die Auffassung der zuständigen Behörde für IT-Sicherheit, des Bundesamtes für Sicherheit in der Informationstechnik (BSI).[192]

6.2. Verbesserung der IT-Sicherheit durch den "Open Source"−Prozess

> «[...] systems that are public are likely to be better scrutinized, and more secure, than systems that are not.»[193]

Die Verbesserung der Sicherheit von Software, von IT-Systemen überhaupt, ist ein Prozess, der etwa nach *Pipkin* folgende Schritte umfasst:[194]

- *«Inspection»*, d.h. die Analyse des bestehenden Sicherheitsniveaus
- *«Protection»*, d.h. die aktive Herstellung eines höchstmöglichen Sicherheitsniveaus
- *«Detection»*, d.h. die Feststellung sicherheitsrelevanter Aktivitäten
- *«Reaction»*, d.h. die angemessene Handlung im Falle einer Sicherheitsgefährdung
- *«Reflection»*, d.h. die kontinuierliche Sicherheitsevaluierung

---

192 Vgl. z.B. **BSI DDOS 2000**.

193 **Schneier 2000**, p. 344.

194 Vgl. **Pipkin 2000**, p xx.

Dabei herrscht noch großes Unverständnis über die Bedeutung dieser essentiellen Schritte − auch unter Anhängern der "Open Source"-Bewegung:

1. Ein höheres Sicherheitsniveau wird nicht automatisch durch die Offenlegung der Wirkungsmechanismen und der Implementierung eines Informationssystems erreicht. Ein Sicherheitsgewinn wird in der Folge eines kontinuierlichen Evaluierungsprozesses und gegebenenfalls einer fehlerbereinigten Re-Implementierung im Lichte der Ergebnisse der Evaluierung erzielt. Der "Open Source"-Prozess unterstützt die genannten Schritte in herausragendem Maße.
2. Ein weitverbreitetes Missverständnis – auch unter "Open Source"-Anhängern – ist, dass bereits die Offenlegung des Quellcodes einer beliebigen Software *xyz* talentierte Entwickler und Nutzer verlocken würde, darin nach Fehlern zu suchen. Dem ist nicht so: Nur wenn Anwender und Entwickler in ausreichendem Maße *Interesse* an der jeweiligen Software haben, ist ihre Kooperation zu erwarten. Hinzu kommt die Erwartung der "Open Source"-Anwender und -Entwickler, einen kooperativen und verantwortlich handelnden Software-Anbieter zu unterstützen. Versagt der Urheber des Quellcodes im Kooperationsprozess, beispielsweise durch ungenügende oder falsche Reaktionen auf die Vorschläge der "Open Source"-Entwickler und -Nutzer, so frustriert er diese und verliert ihre Mitarbeit. Die Schwierigkeiten mit der "Open Source"-Variante des Netscape-Webbrowsers liefern ein beredtes Beispiel für diese These.
3. Ein anderer Faktor, der sich auf die Sicherheit von IT-Systemen auswirkt, ist die Verfügbarkeit von sicherer Software unter betriebswirtschaftlichen Maßstäben. Sichere, praktisch aber unbezahlbare Software wird wahrscheinlich nicht zum Einsatz kommen.[195]

Bezahlbare, sichere Software ist unerlässlich

Eine verantwortungsbewusste IT-Politik für die Informationsgesellschaft muss daher das Ziel der bezahlbaren Versorgung mit qualitativ hochwertiger, sicherer Software verfolgen. Die Förderung der Entwicklung von "Open Source"-Software ist nach gegenwärtigem Wissensstand der einzig gangbare Weg zur Erreichung dieses Ziels. Die Besteuerung des Einsatzes von kostenlos im Internet verfügbarer "Open Source"-Software, wie sie jetzt in Polen praktiziert wird, erscheint unter diesem Gesichtspunkt kontraproduktiv.[196]

Die Verfügbarkeit sicherer Software, das hat die sogenannte "Kryptodebatte" der vergangenen Jahre gezeigt, hängt aber auch von der politischen Unbeeinflussbarkeit der Software-Hersteller zusammen. Aus politischen Gründen unsicher gemachte Software lädt potentielle Angreifer geradezu ein, die Schwächen der Software missbräuchlich auszunutzen.

Die weltweite Entwicklung von "Open Source"-Software hat die Verfügbarkeit modernster Verschlüsselungstechnologien in Europa und Deutschland sichergestellt, obwohl beispielsweise die

---

195 Vgl. **Bartsch 2000**, S. 724.

196 Linux Today v. 20.9.2000, im Internet: http://linuxtoday.com/news_story.php3?ltsn=2000-11-20-001-20-NW-CY-LF.

USA deren Verbreitung aus Eigeninteresse lange Zeit zu unterbinden suchten.[197]

Die folgenden Abschnitte wenden sich nun einzelnen Schritten auf dem Weg zu einer sichereren IT-Infrastruktur zu und untersuchen, welchen Beitrag "Open Source"-Software dazu geleistet hat oder zu leisten vermag.

6.2.1. Sicherheitsgewinn durch Evaluierung der Software

> «Für Microsoft ist Sicherheit eine Frage des Vertrauens − Sun setzt auf Kontrolle.»[198]

Der Dresdner Informatiker *Andreas Pfitzmann*, der wohl führende Spezialist für IT-Sicherheit in der Bundesrepublik, hat im September-Heft der Zeitschrift Datenschutz und Datensicherung zusammen mit anderen den Stand der Fachdiskussion zusammengefasst:

> «Für eine umfassende Sicherheitsuntersuchung ist es erforderlich, das Gesamtsystem zu analysieren, d.h. außer der Anwendungssoftware einschließlich ihres Quellcodes auch die Werkzeuge, die zur Erstellung des Objektcodes verwendet werden, wie Compiler, Betriebssystem und Hardware sowie die gesamte Einsatzumgebung. So können beispielsweise in den Entwicklungswerkzeugen eingebaute trojanische Pferde ihre Schadenswirkung auch dann entfalten, wenn der Anwendungsquellcode davon keine Spuren aufweist [...].»[199]
>
> «Zwar reicht allein die Offenlegung des Codes für Sicherheit nicht aus, jedoch ist sie eine essentielle Voraussetzung für effektive Sicherheitsuntersuchungen [...].»[200]

---

197 Siehe z.B. die Software "Fortify", mit deren Hilfe der Netscape Navigator auf 128-Bit--Verschlüsselung aufgerüstet werden kann. Im Internet: http://fortify.net (30.11.2000).

198 So lautet der Titel eines Artikels in der **Computer Zeitung 10/1997**, S. 9, in dem Microsofts Auffassung beschrieben wird, dass eine echte Sicherheitskontrolle unnötig sei. Konkret bezieht sich die Aussage auf die "ActiveX"-Technologie, die als Konkurrenz zu der plattformunabhängigen Programmiersprache "Java" von SUN Microsystems positioniert wird.

199 **Köhntopp/Köhntopp/Pfitzmann 2000**, S. 511.

200 **Köhntopp/Köhntopp/Pfitzmann 2000**, S. 513.

Es ist herrschende Meinung unter den Experten, dass nur die Offenheit des "Open Source"-Entwicklungsprozesses und der "Open Source"-Software die notwendigen Voraussetzungen liefert, um eine effektive Sicherheitsevaluierung durchführen zu können.

> «There is, of course a difference between a vulnerability and an intrusion point, but that difference is usually only one of degree.»[201]

Anhand des offengelegten Quellcodes ist es Experten möglich, "Open Source"-Software auf Sicherheitsrisiken wie Design- oder Implementierungsfehler und "bösartigen Code" (malicious code) zu untersuchen. Eine Vielzahl der klassischen Software-Sicherheitsrisiken[202] lassen sich durch eine geeignete Code-Evaluierung ausschließen oder aufdecken. Dazu gehören:

- Pufferüberläufe[203] (buffer overflows)
- Logische Bomben[204]
- Parasiten[205]
- Sniffer[206]
- Spoofs[207]
- Trojanische Pferde[208]
- Viren[209]

Aber auch für die Untersuchung proprietärer Software auf Virenbefall oder Trojanische Pferde stellt "Open Source"-Software eine geeignete Umgebung bereit, in der eine weitgehend gefahrlose

---

201 **Cooper et al. 1995**, p. 8.

202 Vgl. **Pipkin 2000**, p. 44ff.

203 Nach Aussage von *Bruce Schneier* waren zwei Drittel der vom CERT (Coputer Emergency Response Team an der Carnegie Mellon Universität) im Jahr 1998 benannten Sicherheitslücken auf Pufferüberläufe zurückzuführen. Vgl. **Schneier 2000**, p. 363.

204 Bestandteil von Software, der bei Erreichen definierter Umstände Schaden anrichet. Vgl. **Pipkin 2000**, p. 356.

205 Software, die unerlaubt Ressourcen des befallenen Systems benutzt. Vgl. **Pipkin 2000**, p. 356.

206 Software, die den Datenverkehr belauscht. Vgl. **Hughes Jr. 1995**, p. 358.

207 Software, die absichtlich falsche Angaben zur eigenen Identität macht. Vgl. **Pipkin 2000**, p. 358.

208 Software, die im Interesse eines Angreifers unzulässige Operationen ausführt und darüber den Anwender täuscht. Vgl. **Pipkin 2000**, p. 358.

209 Software, die sich selbst kopiert und an andere Software anhängt. Oft enthalten Viren auch schädliche Funktionen. Vgl. **Pipkin 2000**, p. 358.

Evaluierung möglich ist. So setzt der bekannte Hersteller von Antiviren-Programmen Symantec auf Linux als sichere Untersuchungsumgebung für den Virenbefall von Software für Microsoft--Betriebssysteme.[210]

Evaluierung setzt Kriterien voraus, nach denen die Sicherheit eines IT-Systems − oder eines Teils davon − zu untersuchen und zu bewerten ist. Um Vergleichbarkeit zu gewährleisten, müssen die Kriterien und Methoden der Untersuchung und Bewertung in der einen oder anderen Form standardisiert und offengelegt sein.

Offene Standards

Offene Standards bieten einen geeigneten Ansatzpunkt für die Erarbeitung von produkt- und anbieterneutralen Evaluierungskriterien. Bei der Entwicklung des Internet und von "Open Source"-Software spielen offene Standards seit jeher eine herausragende Rolle und ihre Einhaltung sorgt für die Interoperabilität und Zuverlässigkeit bei der Kommunikation zwischen den unterschiedlichsten Plattformen.

Einsatz kryptographischer Verfahren

Von besonderer Bedeutung ist die Evaluierung im Zusammenhang mit dem Einsatz kryptographischer Verfahren zur Absicherung der System- und Datenintegrität. Eine Anzahl kryptographischer Verfahren ist seit langer Zeit bezüglich ihrer Qualität untersucht worden. Die von den Experten festgestellte Sicherheit der Verfahren kann jedoch nur gewährleistet werden, wenn auch die Implementierung der Verfahren fehlerfrei erfolgt. Die korrekte Umsetzung von Referenzimplementierungen bildet eine Möglichkeit dazu und kann von Experten mit vertretbarem Aufwand geprüft werden.

Zertifizierung

Anhand der einsehbaren Implementierung ist nicht nur eine Evaluierung, sondern auch eine Zertifizierung und Einstufung nach anerkannten IT-Sicherheitskriterien wie z.B. den *Common Criteria* (CC) oder den *Information Technology Security Evaluation Criteria* (ITSEC) durchführbar. Eine solche Zertifizierung ist geeignet, nicht nur die Sicherheit des jeweiligen Informationssystems zu erhöhen, sondern auch das Vertrauen der Anwender in die Technologie zu fördern.[211] Dieses Vertrauen ist neben der Sicherheit eine Vorbedingung für den Erfolg der Informationsgesellschaft.

---

210 **Kürten 1998**.

211 Vgl. dazu auch: **Köhntopp/Köhntopp/Pfitzmann 2000**, S. 512.

Wiederverwendung

Ein möglicher Ansatz für die Integration sicherer Software stellen Software-Bibliotheken dar, die von Experten geprüft und für korrekt befunden wurden. Software, die solche Bibliotheken unverändert verwendet, weist dann zumindest in diesem Teil der Implementierung mit hoher Wahrscheinlichkeit keine Schwachstellen auf.

Auch aus software-technischer Perspektive scheint ein Einsatz vorgefertigter Module, die als solche geprüft werden könnten, sehr sinnvoll. Damit würde dem akzeptierten Paradigma der Wiederverwendung Rechnung getragen werden:

> «One thing expert designers know not to do is to solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past. When they find a good solution, they use it again and again. Such experience is part of what makes them experts.»[212]
>
> «When experts work on a particular problem, it is unusual for them to tackle it by inventing a new solution that is completely distinct from existing ones. They often recall a similar problem they have already solved, and reuse the essence of its solution to solve the new problem. This kind of 'expert behaviour', the thinking in problem-solution pairs, is common to many different domains, such as architecture [Ale79], economics [Etz64] and software engineering [BJ94]. It is a natural way of coping with any kind of problem or social interaction [NS72].»[213]

Die Forderung nach dem Einsatz solcher «*allumfassenden in "Standard-Software" implementierten [...] Sicherheit*» wurde von *Lessing/Weese* schon vor Jahren erhoben, wenn eine mögliche Realisierung damals auch in weiter Ferne zu liegen schien.[214] Mit "Open Source"-Software ist man diesem Ziel mittlerweile ein großes Stück näher gekommen.

6.2.2. Sicherheitsgewinn durch Implementierung im "Open Source"−Prozess

> «Security is not a product; it's a process. You can't just add it to a system after the fact.»[215]

---

212 **Gamma et al. 1994**, p. 1.

213 **Buschmann et al. 1996**, p. 2.

214 Vgl. **Lessing/Weese 1995**, S. 31.

215 **Schneier 2000**, p. 395.

Die Sicherheit eines Informationssystems hängt von der Sicherheit jeder einzelnen Komponente des Systems ab.

Es wird häufig das Bild einer Kette bemüht, deren Sicherheit so hoch ist wie die ihres schwächsten Glieds.[216] Maximale Sicherheit erreicht man, indem das gesamte Informationssystem nicht nur an den funktionalen Anforderungen ausgerichtet und später um Sicherheitsmerkmale erweitert wird, sondern indem von vornherein die Systemgestaltung gemäß definierter Sicherheitsanforderungen vorgenommen wird.[217]

Der "Open Source"-Entwicklungsprozess ermöglicht es, schon im Entstehungsprozess eines Systems sicherheitskritische Designentscheidungen festzustellen und zu korrigieren. Die Anforderung der verteilten Entwicklung bringt es mit sich, dass die "Open Source"-Software stark modularisiert ist. Eine solche Modularisierung erleichtert den Evaluierungsprozess erheblich: Kleine Codemengen sind leichter überschaubar und überprüfbar.[218] Sicherheitskritische Fehler lassen sich durch kurzfristige Überarbeitung eines einzelnen Moduls schneller beheben.

Die Praxis zeigt, dass die Reaktionszeiten für die Fehlerbehebung im "Open Source"-Bereich sehr kurz sind. Oft handelt es sich nur um Stunden oder Tage, bis fehlerkorrigierte Versionen oder Patches[219] zur Verfügung stehen, wie z.B. im Falle des sogenannten "F0 0F-Fehlers" beim Intel-Pentium-Prozessor. Aus diesem Grunde empfiehlt das BSI den Einsatz von "Open Source"-Software.[220] Im Vergleich dazu benötigen die Anbieter proprietärer, geschlossener Software oft Monate oder Jahre, bis die Fehler behoben werden. Der geschlossene Charakter der Korrekturen bringt es noch dazu mit sich, dass es schwierig ist festzustellen, ob der Fehler *tatsächlich behoben* oder nur *umgangen* wurde.

Eine weiterer Umstand spricht für die breite Verwendung von "Open Source"-Software: Der weitaus größte Teil der geschriebenen Software wird nicht als Produkt vermarktet, sondern innerhalb von Unternehmen für spezifische Aufgaben entwickelt, angepasst und eingesetzt.[221] Der Entwicklungsaufwand ist jedoch *nicht* geringer als bei der Software-Entwicklung zur Vermarktung. Die Kosten für das Kopieren von Software, wie sie bei einer Produktvermarktung anfallen, sind praktisch zu vernachlässigen. Das gilt insbesondere für die elektronische Vermarktung, die immer stärkere Verbreitung findet. [222]

Problematisch ist jedoch die Qualitätssicherung. Bei dem Einsatz von "in-house"-entwickelter Software stehen weitaus weniger Erfahrungen mit möglichen Fehlerquellen zur Verfügung als bei

---

216 Vgl. **Schneier 2000**, Preface.

217 Vgl. **Schneier 2000**, p. 334ff.

218 Vgl. **Köhntopp/Köhntopp/Pfitzmann 2000**, S. 513.

219 Software, die Fehler in anderer Software behebt.

220 Vgl. **BSI DDoS 2000**.

221 Der Anteil der "in-house"-Entwicklungen wird von *Eric S. Raymond* auf ca. 90-95% am gesamten Volumen von Programmen beziffert. **Raymond 1999**, p. 142.

222 Vgl. z.B. **Shapiro/Varian 1999,** p. 20ff.

Standardsoftware. Der Code durchläuft weniger Inspektionsinstanzen, weswegen möglicherweise weniger sicherheitsrelevante Fehler aufgedeckt werden. Durch die zunehmende Vernetzung der PC-Arbeitsplätze innerhalb von Firmen und auch durch deren Anbindung an das Internet entstehen so völlig neue Sicherheitsrisiken.

Der Einsatz sicherheitsgeprüfter "Open Source"-Software könnte sowohl die Entwicklungskosten für Software in Unternehmen und im öffentlichen Sektor verringern als auch die Qualität der eingesetzten Software und damit die Sicherheit der IT-Infrastruktur verbessern: Das Expertenwissen der Sicherheitsfachleute könnte so allgemein verfügbar gemacht werden. Potentiellen Angreifern würde weniger Angriffsfläche zur Verfügung stehen.

Ein weiterer Aspekt ist die "Lebensdauer" von Software. Im Laufe von längeren Zeiträumen gereifte und sicherer gemachte Software stellt eine wertvolle Investition dar, die es zu nutzen gilt. Die Investitionen in die Aus- und Weiterbildung des Personals sollten dabei nicht unterschätzt werden. Kurze Produktlebenszyklen führen im Gegensatz dazu zu hohen "sunk costs", d.h. zu verlorenen Investitionen vor allem in den Bereichen Software-Beschaffung und Aus- und Weiterbildung. Kommen mit einer neuen Software-Version noch höhere Anforderungen an die einzusetzende Hardware hinzu, erreichen die "sunk costs" oft ein beträchtliches Volumen.[223]

Ein grundsätzliches Problem der modernen Software-Entwicklung ist ihre Abhängigkeit von unterstützenden Werkzeugen wie Compilern, integrierten Entwicklungsumgebungen (IDE[224]), Debuggern, Profilern und Software-Bibliotheken. Die Bemühungen eines Software-Entwicklers um sicheren Code werden oftmals durch Sicherheitsprobleme in den Werkzeugen oder im damit generierten Code zunichte gemacht. Zur Entwicklung von "Open Source"-Software stehen eine Reihe von Entwicklungswerkzeugen zur Verfügung, die ihrerseits als "Open Source"- bzw. "Free Software" entwickelt wurden und somit die Voraussetzung zu einer Qualitäts- und Sicherheitsevaluierung erfüllen.[225]

---

223 Zu "sunk costs" vgl. **Shapiro/Varian 1999**, p. 21ff.

224 *Integrated Development Environment* (IDE) = integrierte Entwicklungsumgebung, eine Software, die alle zur Software-Entwicklung benötigten Werkzeuge wie Editor, Compiler und Debugger beinhaltet.

225 Vgl. auch **Köhntopp/Köhntopp/Pfitzmann 2000**.

6.2.3. Sicherheitsgewinn durch Einsatz von "Open Source"−Software

> «Maßnahme 8: Einsatz von Open-Source-Produkten
>
> Für den Fall, dass Schwachstellen neu entdeckt werden, die einen DoS-Angriff ermöglichen oder erleichtern, ist es wichtig, dass diese schnell behoben werden können. Meist werden derartige Schwachstellen in Open-Source-Software wesentlich schneller behoben als in Produkten, deren Quellcode nicht veröffentlicht ist. Häufig können die Veränderungen im Quellcode sogar selbst durchgeführt werden. Daher sollten Open-Source-Produkte bei ähnlicher Leistungsfähigkeit bevorzugt werden (siehe http://linux.kbst.bund.de/).»[226]

Unter dem Eindruck anfänglich des "Free Software"- und später des "Open Source"-Paradigmas ist eine Reihe qualitativ hochwertiger und sicherer Software entwickelt worden. Darunter befinden sich eine Anzahl von Betriebssystemen, deren bekanntestes Linux ist.[227] Als das sicherste Betriebssystem gegenwärtig gilt unter Experten "OpenBSD",[228] ein freier Abkömmling der UNIX-Variante "Berkeley System Distribution".[229]

Neben Betriebssystemen stehen qualitativ hochwertige Editoren, Compiler und Bibliotheken zur Verfügung. Anwendungssoftware gibt es für (fast) jeden erdenklichen Zweck − und das Angebot wächst täglich.

Allerdings wird der Zugewinn an Sicherheit oft mit Abstrichen bei der Ausstattung der Software mit "Features", d.h. Funktionalität, erkauft. Der Verzicht auf zusätzliche Funktionalität verringert die Komplexität der Software und damit einhergehend die Menge des auf Sicherheitslücken zu

---

226 **BSI DDoS 2000**.

227 Weitere "Open Source"-Betriebssysteme sind z.B. OpenBSD (http://www.openbsd.org), FreeBSD (http://www.freebsd.org), FreeDOS (http://www.freedos.org) und GNU Hurd (http://www.gnu.org/hurd/). Es existieren noch mehr als die hier genannten mit jeweils eigenem Einsatzbereich.

228 Vgl. u.a. **Loshin 2000**: «The stated goal of the OpenBSD group is to make OpenBSD "Number one in the industry for security." Many believe it has reached this goal already: it will soon be three years since a successful remote root attack on OpenBSD has been reported.» **Koerner 2000**: «"OpenBSD is probably one of the most secure operating systems out there," says Chris Brenton, author of Mastering Network Security. "The crew does a fantastic job of locking down and being responsive when vulnerabilities are found." Such a good job that the U.S. Department of Justice uses 260 copies of OpenBSD to store and transmit its most sensitive data.» S. dazu im Internet: http://www.openbsd.org/security.html (18.10.2000).

229 Zur Geschichte der Berkeley System Distribution vgl.: **McKusick 1999**.

prüfenden Codes. Eine Zunahme an Komplexität zieht in der Regel eine Verschlechterung der Sicherheit eines Systems nach sich,[230] was für einen Verzicht auf − unnötige − Komplexität[231] spricht.

Niedrige Migrationskosten bei "Open Source"-Software

Aus betriebswirtschaftlicher Perspektive ist der Einsatz von sicherer "Open Source"-Software, wie z.B. OpenBSD, auch deshalb attraktiv, weil die Migrationskosten zur Erreichung des höheren Sicherheitsniveaus gering sind:

- Lizenzkosten für Betriebssystem und Anwendungssoftware entfallen. Die Kosten für Distributionsmedien sind sehr gering.[232]
- Kosten für teure Hardware entfallen aufgrund der sparsamen Hardware-Anforderungen von "Open Source"-Software.
- Wartungskosten entfallen weitgehend,[233] da diese durch die "Open Source"-Entwicklung von der Entwicklergemeinde getragen werden.
- Die Anpassungskosten sind gering, da bei Vorhandensein des Quellcodes der Anpassungsauftrag im Ausschreibungsverfahren vergeben werden kann, wodurch Kostenvorteile erzielbar sind.

Die genannten Punkte sind in Zeiten knapper Kassen der öffentlichen Haushalte von erheblicher Relevanz bei der Kalkulation für die Absicherung der öffentlichen IT-Infrastruktur.[234]

Das Kostenargument hat noch mehr Gewicht, wenn es nicht um betriebliche Ausgaben, sondern um den Einsatz sicherer Software im Privatbereich geht. Professionelle, proprietäre Software zur Sicherung des privaten Computers ist für den größten Teil der Nutzer schlicht unbezahlbar. Die Kosten dafür liegen in Bereichen von bis zu mehreren Tausend DM und damit außerhalb des normalen privaten Budgets.

---

230 Vgl. **Schneier 2000**, p. 354f.: «Complexity is the worst enemy of security. This has been true since the beginning of computers, and is likely to be true for the foreseeable future. [...] The first reason is the number of security bugs. [...] As the complexity of the software goes up, the number of bugs goes up. And a percentage of these bugs will affect security, and not always in tangible ways.»

231 *Bruce Schneier* bringt als Beispiel für eine unnötige Komplexität die Tabellenkalkulation Microsoft Excel 97. Jede Kopie des Programms beinhaltet − versteckt − einen Flugsimulator, der in keinem Falle einen Beitrag zur Lösung der Aufgaben leisten kann, für die eine Tabellenkalkulation eigentlich gedacht ist. Vgl. **Schneier 2000**, p. 354.

232 Beispielsweise kann man die aktuelle "OpenBSD"-Distribution (2.7) für 69,95 DM (Preis: Oktober 2000) im Handel erwerben.

233 Wartungskosten entfallen nicht völlig, da die Verantwortung für das Einspielen eines vom Hersteller oder Entwickler online zur Verfügung gestellten Updates der Software (zur Beseitigung eventueller Sicherheitslücken) dem Anwender obliegt. Darin gibt es aber keinen großen Unterschied zu proprietärer Software. Bei letzterer kommen ggf. noch separate Lizenzkosten für das Update hinzu.

234 Vgl. im Internet: http://linux.kbst.bund.de/ (26.11.2000).

6.2.4. Sicherheit durch Anbieterunabhängigkeit

Eines ist klar: Dort, wo es nur einen Monopolisten als Anbieter gibt, kann es keinen Wettbewerb um mehr Sicherheit, um preiswertere Sicherheit geben. Die Preise für sicherheitskritische Software werden nicht im Markt reguliert werden und über dem Niveau liegen, das als für alle bezahlbar gelten kann.

Ein Beispiel ist das RSA-Verfahren[235], auf das durch Patentschutz faktisch ein Monopol bestand. Nach Auslaufen des Patentschutzes für RSA am 20. September 2000 ist mit einer Vielzahl von Anbietern zu rechnen. Die Preise für proprietäre Software mit RSA-Verschlüsselung werden sinken und bei "Open Source"-Software ist mit breitem Einsatz des RSA-Verfahrens zu rechnen:

> «The big news in security is the expiration of the RSA patent, which, along with the U.S. government's relaxation of export controls, means we can use open-source security tools such as OpenSSL everywhere. No more paying tribute to the RSA bandits to use their patent. That's great news, and we expect the Linux distributions to start integrating strong crypto everywhere.»[236]

Sicherheit bedeutet mehr als nur Schutz vor Systemeinbrüchen, Systemausfällen und Schäden. In diesen Bereich fallen auch "Investitionssicherheit" und "Zukunftssicherheit".

Die Anwender von "Open Source"-Software verfügen über den Quellcode ihrer geschäftskritischen Software und die Rechte zu dessen Bearbeitung. Damit sind sie zur Verbesserung und Weiterentwicklung der Software nicht mehr auf die Leistung eines einzelnen Anbieters beschränkt, der nach Maßgabe seiner Unternehmensstrategie entscheidet, ob und wie er ein Produkt pflegt. Im Fall von proprietärer Software ohne Quellcode gibt es auf Seiten des Nutzers praktisch keine Möglichkeit, zur Pflege der Software auf einen herstellerunabhängigen Dienstleister auszuweichen.

Eine Weiterentwicklung im Sinne der Vorstellungen des Nutzers ist nur durch "Open Source"-Software praktisch (Quellcode) und urheberrechtlich ("Open Source"-Lizenz) abgesichert. Bei proprietärer Software ist die Weiterentwicklung, und damit die Investitions- und Zukunftssicherheit bezüglich der Software unsicher, da sie von den betriebswirtschaftlichen Interessen des Anbieters abhängig ist.

---

235 Das RSA-Verfahren, benannt nach seinen Erfindern *Rivest*, *Shamir* und *Adleman*, ist das wichtigste Verfahren für die asymmetrische Verschlüsselung, die z.B. bei digitalen Signaturen zum Einsatz kommt.

236 **Marti 2000**.

Inzwischen haben auch einige IT-Dienstleister, darunter das Branchenschwergewicht IBM,[237] aber auch etliche kleine und mittelständische Unternehmen[238] im "Open Source"-Umfeld Chancen zur Vermarktung ihrer Leistungen erkannt und bieten Lösungen, Entwicklung, Beratung und Schulung an.

6.3. Reduzierung der IT−Sicherheit durch Recht?

6.3.1. Software-Lizenzen und Software-Qualität

Es ist denkbar − aber noch nicht ausreichend untersucht −, dass das bestehende Schutzsystem für Software, das Schutz überwiegend durch das Urheberrecht herstellt, eine akzeptable Qualität der Software verhindert.

Der rechtliche Zusammenhang zwischen Software und IT-Sicherheit ergibt sich aus dem exklusiven Bearbeitungsrecht, das dem Schöpfer des Werkes vorbehalten ist, und dem Verbot des "Reverse Engineering" ("Dekompilierung"). Bei proprietärer Software wird dieses Recht in der Regel, d.h. in der Praxis, nicht lizenziert.

Dieser Einschränkung unterliegen auch Bearbeitungen zur Fehlerbeseitigung, wenn die Fehler die bestimmungsgemäße Benutzung der Software nicht wesentlich einschränken. Das Urheberrecht verbietet − mit den Ausnahmen zur Herstellung eines Zustandes der bestimmungsgemäßen Nutzung (§69d UrhG) und der Interoperabilität (§69e UrhG) − ausdrücklich das "Reverse Engineering" von urheberrechtlich geschützter, proprietärer Software.[239] Führt die Fehlerbeseitigung dazu, dass ein wichtiger Teil der Software nicht mehr funktionsfähig ist, muss sie unterbleiben, da sie dann nicht mehr als notwendig gilt. Ob der in der Folge der Fehlerbeseitigung nicht mehr funktionsfähige Teil ein "wichtiger" Teil ist, hängt von der Sicht des Anbieters und nicht des Anwenders ab. Fehler bewirkende Kopierschutzmaßnahmen dürfen jedenfalls nicht deaktiviert werden.[240]

Unter Umständen kann das Recht zur Fehlerbeseitigung auch vertraglich abdingbar sein, d.h. der Hersteller der Software kann den Nutzer vertraglich darauf festlegen, die Fehlerbeseitigung nur durch den Hersteller (oder einen von ihm Beauftragten) durchführen zu lassen. Ob eine solche

---

237 Vgl. z.B. das Interview mit *Irving Wladawsky-Berger,* IBM Vice President Technology and Strategy, zu der "Open Source"-Strategie von IBM, in **Linux Enterprise 3/2000**, S. 16ff: «*Für uns ist Linux so bedeutsam, weil es für Applikationen das bedeutet, was das Internet für Netzwerke bedeutet hat.*»

238 Genannt seien beispielhaft die Firmen Linux Information Systems AG (http://www.linux-ag.com), München, und Innominate (http://www.innominate.de), Berlin.

239 Vgl. **Fromm/Nordemann 1998**, S. 491ff, §69d Rn. 1-6, §69e Rn. 1-5: «Der Begriff der Notwendigkeit ist dabei restriktiv auszulegen.» Vgl. auch **Fromm/Nordemann 1998**, §69d Rn. 3, S. 492, bzw. **Hoeren/Schumacher 2000**.

240 Vgl. **Fromm/Nordemann 1998**, §69d Rn.3, S. 492.

Vertragsregelung in Notfällen greifen würde, ist zweifelhaft. Im Übrigen liegt die Darlegungs- und Beweislast für die Notwendigkeit einer Dekompilierung beim Nutzer, nicht beim Anbieter.

Der Einbau von zusätzlichen Elementen in die Software zur Erhöhung der IT-Sicherheit ist als Verbesserung der Software urheberrechtlich unzulässig, wenn sie nicht ausdrücklich in der Lizenz gestattet wird.[241] Betrachtet man die geschäftsüblichen Lizenzverträge für Standardsoftware, so wird nirgends ein Recht zur sicherheitstechnischen Nachrüstung der Software eingeräumt. Die Lizenznehmer proprietärer Software dürfen Software nicht bearbeiten, um sie an die eigenen Sicherheitsbedürfnisse anzupassen, selbst wenn ihnen Sicherheitslücken bekannt werden.

Folgt man diesen Überlegungen, erscheint folgende **These** plausibel:

> **Das Urheberrecht setzt der Qualitätsverbesserung bei Software (zu) enge Grenzen. Im Falle proprietärer Software unterstützt das Urheberrecht die Herstellung und den Vertrieb unsicherer Produkte. Jeder weitergehende Schutz für Software, der die Belange der IT--Sicherheit nicht angemessen berücksichtigt, wird die ohnehin prekäre Situation weiter verschärfen.**

Sollte sich diese These belegen lassen, muss man fürchten, dass sich eine Prognose von *Bruce Schneier*, einem der renommiertesten internationalen Experten der IT-Sicherheit, bewahrheiten könnte: ***«[T]he laws will reduce security in the long run.»***[242]

In diesem Fall könnte das beschriebene Dilemma der IT-Sicherheit nur überwunden werden, wenn das Urheberrecht auf geeignete Weise geändert wird.

6.3.2. Die Haftungslücke bei Software

> «Software manufacturers don't have to produce a quality product because they face no consequences if they don't. [...] And the effect of this for security products is that manufacturers don't have to produce products that are actually secure, because no one can sue them if they make a bunch of false claims of security.»[243]

---

241 Vgl. **Fromm/Nordemann 1998**, §69d Rn. 3, S. 492.

242 **Schneier 2000**, p. 346.

243 Vgl. **Schneier 2000**, p. 365.

Die Einordnung von Software als urheberrechtlich/Copyright-geschütztes geistiges Eigentum hat ein erhebliches Problem mit sich gebracht: eine *Haftungslücke*.

Betrachtet man die klassischen Gegenstände des Urheberrechts (einschließlich des Copyright), so haben sie eines gemeinsam: Sie repräsentieren einen "ästhetischen Gehalt", der sie einer inhaltlichen Bewertung faktisch entzieht. Das Werk verkörpert die Vorstellungen seines Schöpfers, der man zustimmen oder die man ablehnen kann. Man kann den Schöpfer für gewöhnlich aber nicht für seine Vorstellungen haftbar machen (Ausnahmen in Fällen von Beleidigung usw. bestätigen die Regel). Das Urheberrecht selbst kennt keine "Schöpferhaftung".

Dies ist verhältnismäßig unproblematisch, solange ein Werk nur *als solches* "genossen" wird. Wenn sich in der Folge des Werksgenusses bei Literatur, Musik, Fotos etc. ein Schaden einstellt, so dürfte das eine große Ausnahme sein.

Im Unterschied dazu ist Software nicht einfach nur ein Text. *Sie verhält sich* in bestimmter Art und Weise. Wer ein Programm kauft, will es nicht lesen, *das Programm soll etwas tun.* Hierin unterscheiden sich Programme nicht von anderen Maschinen. *Software ist eine Maschine*:

> «Thinking of software as a machine makes it clear that source code is the medium in which a program is created, even though the value in a program, as with other machines, lies in it's behavior.»[244]

Das muss natürlich Konsequenzen für den Schutz haben. Die herrschende Meinung der akademischen Öffentlichkeit der USA hat diese Konsequenzen in einem berühmt gewordenen Symposium an der Columbia Law School im Frühjahr 1994 erörtert. Dabei scheint der Ausgangspunkt überwiegend unstrittig:

> «The predominantly functional nature of program behavior and other industrial design aspects of programs precludes copyright protection, while the incremental nature of innovation in software largely precludes patent protection.»[245]

---

244 **Manifesto 1994**, p. 2323.

245 **Manifesto 1994**, p. 2333.

Das Urheberrecht (und ebenso das Copyright) ist also ein denkbar schlechter Ort für die rechtliche Regelung des Schutzes von Software.

Diese problematische rechtliche Einordnung von Software hat dazu geführt, dass man dem Problem der *Sicherheit von Software als Produkt* zu wenig Aufmerksamkeit geschenkt hat. Niemand würde auf den Gedanken kommen, etwa Goethes "Faust" nach Fehlern abzusuchen.

Da Fehler natürlich immer wieder vorkommen, greift man auf Hilfskonstruktionen wie das Kaufrecht,[246] das Produkthaftungsrecht[247] oder die allgemeine Sorgfaltspflicht zurück.

Bei Standardsoftware haben sich diese Konstruktionen als praktisch wirkungslos erwiesen, so dass kein großer Software-Hersteller bisher für die Mängel der von ihm entwickelten und verbreiteten Software haften musste. Da es keine wirksamen Strafen gibt, besteht auch keine Anreiz, etwas gegen die bekannten Schwächen zu unternehmen.[248]

Über Jahre hinweg sind immer wieder erhebliche Mängel bei den Produkten der großen Software-Hersteller zu verzeichnen, wobei insbesondere die Firma Microsoft − weltgrößter Software-Hersteller − negativ auffällt:

---

246 So z.B. BGH Urteil vom 4.11.1987 - VIII ZR 314/86, CR 1988, S. 124ff. Kritisch zu dieser Einschätzung: **Ruppelt 1988**.

247 *Bartsch* plädiert dafür, dass Microsoft für den Schaden, der durch den "I LOVE YOU"-Virus verusacht wurde, nach §3 ProdHaftG bzw. §823 BGB haften muss, da die ausgelieferte Software sicherheitstechnisch nicht dem Stand der Technik und somit auch nicht den berechtigten Sicherheitserwartungen der Anwender genügte. Der Virus zerstörte bekanntlich Daten und schädigte somit das Eigentum der Betroffenen (**Bartsch 2000**).

Es kann bezweifelt werden, dass diese Argumentation tragfähig ist. Viele Software-Hersteller arbeiten mittlerweile mit Qualitätsmanagementsystemen. Dadurch kommen sie ihrer gwöhnlichen Sorgfaltspflicht nach und die Produktqualität konnte verbessert werden. Fehlerfrei, das hat die Praxis gezeigt, wird die Software dadurch jedoch nicht. Herrschende Meinung ist, dass (komplexe) Software fehlerhaft ist. Wenn die Sicherheitsmängel der Software aus *unvermeidbaren* Fehlern resultieren, kann jedoch kein Anwender davon ausgehen, dass die von ihm eingesetzte Software sicher ist. Eben die Sicherheitserwartung des durchschnittlichen Anwenders aber ist es, an der sich der Fehlerbegriff des §3 ProdHaftG ausrichtet. (Vgl. **Rothe 1993**, S. 311) Ohne Sicherheitserwartung keine Haftung, so ließe sich folgern.

248 Viele der Computervirus-Attacken des Jahres 2000 waren, da sind sich die Sicherheitsexperten einig, nur durch lange bekannte und vom Software-Hersteller nicht beseitigte Schwächen der Programme möglich. Auch der jüngste Einbruch (November 2000) in das firmeninterne Netzwerk von Microsoft erfolgte unter Ausnutzung einer seit März 2000 bekannten Sicherheitslücke. S. dazu im Internet: http://www.heise.de/newsticker/data/pab-06.11.00-000/ (6.11.2000).

«Anwender müssen mit einem Restrisiko leben. Die Virenjäger haben oft nur Platzpatronen geladen.»[249]

«Netzwerk-Anwendern droht Gefahr. OLE: Freie Bahn für Hacker.»[250]

«Qualitätskontrollen der Hersteller lassen zu wünschen übrig. Selbst Originaldisketten sind oft schon mit Viren verseucht.»[251]

«Die Microsoft-Implementierung des PPTP zeichnet sich durch Sicherheitslücken aus.»[252]

«Win-NT-File-System birgt Virengefahr.»[253]

«Win 2000 erlaubt jedem den Log-in.»[254]

Die Präventionsfunktion der gesetzlichen Haftungsregelungen hat bei Software − jedenfalls bisher − versagt. Das Risiko der Software-Benutzung liegt ganz allein beim Anwender. Eine Kompensation steht ihm für diese einseitige Risikoübertragung jedoch nicht zu. Das Gesetz verbietet ihm sogar, Mängel zu beseitigen, auch wenn er dazu in der Lage wäre.[255]

Die Reaktion der Benutzer auf diese bekannte Situation ist einerseits Resignation[256] und andererseits ein fehlendes Qualitätsbewusstsein. *"Software ist fehlerhaft"*, so lautet eine weitverbreitete Meinung.[257] Da die Software undurchschaubar ist, kann man als Anwender sowieso nicht wirklich etwas gegen fehlerhafte Software tun, so die Schlussfolgerung. Aus Fehlern in der Software resultieren aber häufig auch Sicherheitsmängel. Im Ergebnis stellt sich, und das ist in gewisser Weise fatal,

---

249 **Computer Zeitung 46/1993**, S. 20.

250 **Computer Zeitung 6/1994**, S. 12.

251 **Computer Zeitung 11/1996**, S. 2.

252 **Computer Zeitung 24/1998**, S. 13.

253 **Computer Zeitung 38/2000**, S. 8.

254 **Computer Zeitung 46/2000**, S. 1.

255 Die eigenhändige Fehlerbeseitigung ist nur sehr eingeschränkt zulässig. Vgl. **Fromm/Nordemann 1998**, UrhG §§69d-e, S. 491ff. In den USA stellt sich die Situation vergleichbar dar. Vgl. **Rosenoer 1997**, p. 22ff: «Repair Deemed Unlawful.»

256 Vgl. **Bartsch 2000**, S. 724: «Viele Geschädigte werden meinen, dass der Schaden schicksalhaft eingetreten sei und ein Ersatzanspruch nicht bestehe.»

257 Vgl. **Bartsch 2000**, S. 721; **Gorny 1986**, S. 674; **Lesshafft/Ulmer 1988**, S. 813ff; **Bömer 1989**, S. 361; **Taeger 1996**, S. 256.

kein wirkliches Sicherheitsbewusstsein auf Seiten der Anwender ein.[258]

Ein Seiteneffekt der *de-facto-Haftungsentlastung* für Software-Hersteller besteht in deren finanzieller Besserstellung gegenüber anderen Produzenten.

Während die Hersteller von Autos, Haushaltsgeräten, Computern usw. mit Kosten für Sicherheitstests, Produktnachbesserungen oder gar -rückrufe kalkulieren und entsprechende Rücklagen oder Versicherungen finanzieren müssen, kann in der Software-Branche − weitgehend − darauf verzichtet werden. Diese indirekte Subventionierung ist angesichts der exorbitanten Gewinnzuwächse bei großen Software-Anbietern nur schwer zu rechtfertigen. Zumal durch die geringere Fehleranfälligkeit vergleichbarer "Open Source"-Produkte in der Praxis der Nachweis erbracht worden ist, dass Software sicherer sein kann, wenn der Qualitätssicherung eine angemessene Priorität im Entwicklungsprozess zukommt. In jedem Fall trägt die Haftungsentlastung nicht zur Motivierung eines besseren Sicherheitsbewusstseins bei.

Die ungleiche Risiko- und Haftungszuweisung, die aus der hier beschriebenen Haftungslücke resultiert, sollte Anlass zum Nachdenken geben.

> *Es erscheint uns notwendig, die Risikoverteilung bei Software zu überdenken und die Interessen der Anwender angemessener zu berücksichtigen, als das heute geschieht.*

6.4. Zusammenfassung

> «Windows 2000 is not secure enough to use on internet connected servers, according to a senior security analyst at Gartner. [...] Pescatore also claimed that "managed" Linux will become the most secure option in the next five years.»[259]

Die klassischen, kommunikationshinderlichen und sicherheitsproblematischen Grenzen sind für die "Open Source"-Bewegung gefallen:

---

258 Auf der Anwenderseite, so führt *Voßbein* aus, hat die Erfahrung mit der üblichen Unzuverlässigkeit der Computersysteme zu einem bedenklichen Abbau des Bewusstseins für die Eigenverantwortung geführt. So verliert die Systemsicherheit ihr wichtigstes Standbein. Hinzu kommt das Vorgehen der Hersteller, «*Produkte so schnell wie möglich auf den Markt zu werfen, um einen Vorsprung auf dem Markt zu erzielen, was zu tendenziell unsicheren Produkten aufgrund fehlender Durchprüfung und sorgfältiger Testung führt.*» Vgl. **Voßbein 1995**, S. 48.

259 **Ticehurst 2000**. Danach gehen Marktforschungsunternehmen davon aus, dass Linux in Zukunft als das sicherste Betriebssystem fungieren wird.

- Es gibt keine Weisungshierarchien, die über die Verwendung des verfügbaren Wissens befinden müssten. "Verhandlungskosten" können dadurch erheblich reduziert werden.

  → **Sicherheit ist nicht mehr davon abhängig, "Insider" einer bestimmten Hierarchie zu sein. Sichere Informationstechnologie ist für jeden erreichbar und machbar geworden.**

- Es gibt kein geheimes Wissen, aus dem einer Seite ein Vorteil bzw. Nachteil erwachsen würde. Der Wegfall von Lizenzkosten ermöglicht die Verwendung von Expertenwissen in Bereichen, in denen es üblicherweise zu teuer wäre oder als zu teuer eingeschätzt wird.[260]

  → **Für die Absicherung von Informationssystemen ergibt sich daraus eine kostenneutrale Verfügbarkeit von Expertenwissen, was stimulierend auf den "Wettbewerb" um die Verbesserung der Sicherheit wirkt.**

- Das Wissen wird transparenter und leichter vergleichbar, die Qualität des Wissens ist in gewissem Umfang "messbar" geworden. Der Stand dessen, was als sicher gelten kann, ist bekannt.[261]

  → **Im IT-Sicherheitsbereich bedeutet das einerseits eine mögliche Qualitätsverbesserung und andererseits eine verbesserte Grundlage für die Vertrauensbildung, die als Schlüsselfaktor für den Erfolg der New Economy angesehen wird.**

- Missverständnisse werden durch einheitliche Darstellung, Protokolle und Sprachen reduziert. Insbesondere die Ausdrucksform des Quellcodes mit seiner Klarheit hilft, Missverständnisse aufzulösen.

---

260 Wie *Lessing/Weese* betonen, wird durch Investitionen in die Systemsicherheit aus betriebswirtschaftlicher Sicht kein Gewinn erzielt. Die möglichen Einsparungen durch nicht eingetretene Schäden an der IT-Infrastruktur lassen sich betriebswirtschaftlich nur schwer oder gar nicht kalkulieren. Aus diesem Grunde unterbleiben Sicherheitsmaßnahmen, die in anderen Industriezweigen als unverzichtbar angesehen werden. Wenn ein Schadensfall eintritt, wird er von den Betroffenen oft verschleiert, «*weil die Publizierung einem Eingeständnis gravierender Managementfehler gleichkommt.*» Vgl. **Lessing/Weese 1995**, S. 23.

261 Die Bekanntheit der Sicherheitsrisiken würde auch die Voraussetzungen für ein umfassendes *Risikomanagement* liefern, wie es in anderen Bereichen des täglichen Lebens schon lange etabliert ist. Man denke nur an Kranken- und Sozialversicherungen, Sicherheits- und Katastrophenpläne in der chemischen Industrie oder die Vorgaben für Ad-hoc-Informationen von börsennotierten Unternehmen. Im wichtigen Informationstechnologiesektor ist ein vergleichbares Risikomanagement noch nicht gegeben. Zwar gibt es einzelne Unternehmen, die ein solches Risikomanagement praktizieren, doch schon in der öffentlichen Verwaltung sind es nur noch einzelne Bereiche, die systematisch gesichert werden. Aus volkswirtschaftlicher Perspektive kann von einem Risikomanagement in der Informationstechnologie nicht die Rede sein.

→ **Bezogen auf die IT-Sicherheit bedeutet das eine erhebliche Verbesserung, sind doch Missverständnisse auf unterschiedlichster Ebene der Systementwicklung (Aufgabenspezifikation, Design, Implementierung, Test) eine der Hauptursachen für Sicherheitsmängel.**

- Das Wissen ist nicht mehr an Lokalitäten gebunden. Problemstellungen können ungehindert den jeweiligen Experten zur Kenntnis und Lösungen ungehindert zu den Orten der Probleme gelangen.

→ **Gerade im Sicherheitsbereich ist der Aspekt der möglichst kurzen Reaktionszeiten von herausragender Bedeutung. Nur so kann im Falle bekannt gewordener Sicherheitslücken ausreichend schnell reagiert werden, um einen Schaden zu verhindern oder einzudämmen.**

Köhntopp/Köhntopp/Pfitzmann haben das Potential von "Open Source"-Software für einen Sicherheitsgewinn in einer Tabelle folgendermaßen resümiert:[262]

Möglicher Sicherheits-/Vertrauensgewinn	***Open-Source-Eigenschaft***	***Einschränkungen und Grenzen***	***Lösungen und Schutzmaßnahmen***
Quellcode-Review möglich durch beliebige unabhängige Experten	Offenlegung	keine Garantie, dass tatsächlich ein vollständiger Review durch unabhängige Experten durchgeführt wird und dass der Code verständlich wird	Verwenden formalisierter Verfahren für Review und Evaluation; Berücksichtigung allgemeiner Grundsätze der Software-Entwicklung
System basiert nicht auf „Security by Obscurity"	Offenlegung	geht nur bei „reifen" Sicherheitstechniken wie Kryptographie	fragwürdige und „unreife" Sicherheitstechniken, wie z.B. Watermarking vermeiden
keine trojanischen Pferde im System	Offenlegung	nur insoweit das *vollständige* System (incl. Erzeugung des Objektcodes) offengelegt und entsprechend evaluiert wurde	Review und Evaluation des gesamten Systems; Open Source auch für alle verwendeten Werkzeuge
schnelle Fehlerbehebung möglich durch Verbreitung von Patches; Anpassung an eigene Gegebenheiten	Offenlegung, Änderungs- und Verbreitungsmöglichkeit	dadurch Möglichkeit, dass sich neue Fehler einschleichen oder dass interne Angreifer in der eigenen Version flexibler trojanische Pferde einbauen	Test vor Installation des Patches; Kapselung der Produktionsversion gegen unbefugte Veränderung (z.B. durch digitale Signaturen und Zertifikate)
Nutzer kann selbst einsteigen und sich an der Entwicklung und der Qualitätssicherung beteiligen	Offenlegung, Änderungs- und Verbreitungsmöglichkeit	nur für Personen mit Programmierkenntnissen praktikabel	Bildungskampagnen für den Bereich IT
keine Abhängigkeit von einzelnen Herstellern	Offenlegung, Änderungs- und Verbreitungsmöglichkeit	keine Vertragsbeziehung, keine Haftung und Gewährleistung der Autoren	Wartungs- und Distributionsvertrag

262 **Köhntopp/Köhntopp/Pfitzmann 2000**, S. 512.

Möglicher Sicherheits-/Vertrauensgewinn	***Open-Source-Eigenschaft***	***Einschränkungen und Grenzen***	***Lösungen und Schutzmaßnahmen***
Qualitätssicherung durch persönliche Motivation der Entwickler; Reputation statt Zeitdruck durch Marketing-zwänge	Offenlegung, Kostenfreiheit, Verbreitung	Ausrichtung nur auf persönliche Interessen der Entwickler (oft mehr algorithmischer Kern als Oberflächengestaltung)	Ausrichtung auf Kunden durch Distributoren (z.B. Oberflächen, Support, Wartung, Hotlines); Gewährleisten von Authentizität bei Code und Patches
großer Fundus von bewährtem Code, der in neuen Projekten verwendet werden kann	Verbreitung		

Fasst man das in den letzten Abschnitten Gesagte zusammen, lässt sich feststellen, dass der "Open Source"-Software-Entwicklungsprozess große Potentiale hat, die Sicherheit sowohl der privaten, der betrieblichen als auch der öffentlichen IT-Infrastrukt1ur entscheidend zu verbessern. Sichere "Open Source"-Software ist bereits jetzt kostengünstig verfügbar und wird vielfältig genutzt.[263]

Unsere Untersuchungen belegen:

> *Ohne eine Veränderung der Regularien des gewerblichen Rechtsschutzes ist ein Mehr an Sicherheit wahrscheinlich nicht zu erzielen.*

Entsprechend der eingeschränkten Zielsetzung des Kurzgutachtens beschränken wir uns hier auf Empfehlungen für die Novellierung des Patentrechts (Kapitel 7).

263 Vgl. z.B. das Interview mit *Siegmar Mosdorf*, Parlamentarischer Staatssekretär im Bundeswirtschaftsministerium, zur IT-Sicherheitspolitik der Bundesregierung, in **Linux Enterprise 3/2000**, S. 19ff: «In sicherheitsrelevanten Bereichen, wie z.B. im "Informationsverbund Bonn Berlin" arbeiten wir vorrangig mit Linux-Servern.», S.20; «Generell kann die Sicherheit durch freie Software (Open Source) erhöht werden, weil hier die Programmierung bis ins Detail offen liegt und damit überprüft werden kann. Ich bin überzeugt, dass die Open Source Entwicklung den europäischen Ansatz im Informationszeitalter bilden kann. Außerdem wird damit ein permanenter Innovationsprozeß unterstützt.», S.21.

7. Bewertung patentrechtlicher Konsequenzen

7.1. Materielles Patentrecht

Die derzeitige Diskussion um die Änderungen des materiellen Rechtes der Patentierbarkeitsvoraussetzungen verfehlt ihr Ziel. In der Praxis sind Patente, die Auswirkungen auf den Software-Markt haben können, anhand fixer Kriterien nicht zuverlässig und sicher von anderen Patenten abzugrenzen.

Dies hängt wesentlich mit der seit Jahrzehnten im Patentwesen gängigen und, im Hinblick auf die stetige Verwissenschaftlichung des Erfindungsprozesses, auch notwendigen Praxis zusammen, in Patentansprüchen den geschützten Gegenstand nicht mehr nur durch konkrete konstruktive Merkmale, sondern durch die Angabe einer Funktionalität zu kennzeichnen (sogenannte "funktionale Merkmale"). Funktionale Merkmale können vielfach, aber nicht immer, nicht nur durch spezialisierte Schaltungen oder andere Einrichtungen, sondern auch durch Universalrechner, also mittels Software, realisiert werden.

Für die in dieser Situation erkennbaren politischen Handlungsalternativen folgt hieraus folgende Bewertung:

- Das dauerhafte Festhalten am derzeitigen Wortlaut von §1 PatG bzw. von Artikel 52 EPÜ, verändert die Lage für das "Open Source"-Modell nicht.

  Die zuständigen Gerichte in der Bundesrepublik Deutschland bzw. die Beschwerdekammern des Europäischen Patentamtes haben die Abgrenzung des Bereiches patentfähiger Erfindungen gegenüber solchen Neuerungen, denen der Patentschutz zu versagen ist, in vertretbarer Weise auf das gewohnheitsrechtlich gefestigte und nunmehr auch in Art. 27 des TRIPS-Abkommens kodifizierte Erfordernis der Technizität des jeweiligen Anspruchsgegenstandes gestützt. Die Formulierung der Ausschlusstatbestände in §1 PatG bzw. in Artikel 52 EPÜ ist aufgrund begrifflicher Unklarheit gesetzgeberisch völlig misslungen. Eine − von mehreren möglichen − Interpretationen besagt, dass Datenverarbeitungsprogramme als linguistische Konstrukte in ihrer jeweiligen Ausdrucksform vom Patentschutz ausgenommen sind. Dies wäre jedoch eine Trivialität, da der Streit um die Zulässigkeit informationstechnischer Algorithmenpatente geht. Die Beibehaltung des Status quo bedeutet für "Open Source"-Projekte ein unverändert hohes Patentverletzungsrisiko.
- Es sind Ansätze vorstellbar, die materiellen Patentierbarkeitsvoraussetzungen in §1 PatG bzw. in Artikel 52 EPÜ zu modifizieren: Die Erteilung von Patenten auf Patentansprüche, bei denen klar erkennbar ist, dass sie sich auf computer-implementierte Erfindungen beziehen, könnte abgelehnt werden. Das Risiko einer Patentverletzung im Ambivalenzbereich der (auch) computer-implementierbaren Patentgegenstände kann so jedoch nicht ausgeschlossen werden. Das betrifft auch "Open Source"-Projekte.
- Schließlich wäre vorstellbar, die materiellen Patentierbarkeitsvoraussetzungen derart zu modifizieren, dass die Erteilung von allen Patenten ausgeschlossen ist, die sich auf computer--implementierbare Erfindungen beziehen.

  Ein derartiger Ansatz würde das Risiko von Patentverletzungshandlungen durch Gebrauch von Datenverarbeitungsprogrammen nicht nur im Bereich der "Open Source"-Software ausschalten. Er scheitert jedoch aus anderen Gründen:

  Zum einen ist es zum Zeitpunkt der Patentprüfung kaum möglich, die zukünftige technische Entwicklung zu antizipieren. Es kann keine verlässliche Prognose darüber abgeben werden, welche Anspruchsgegenstände im Ambivalenzbereich einen Einfluss auf den Software-Markt nehmen könnten und welche nicht. Zum anderen ist es auch nicht vorstellbar, die Gesamtheit aller Patentgegenstände mit potentiellen Auswirkungen auf Software vom Patentschutz

auszuschließen. Hierbei würde der Patentschutz einem zu großen Kreis von Erfindungen versagt werden.

Keine Lösungen für "Open Source"-Software auf der Basis des geltenden Patentrechts

Das Problem, das das "Open Source"-Modell mit dem Patentrecht hat, kann daher − unter den gegebenen Rahmenbedingungen − augenscheinlich nicht durch eine Reform der materiellen Patentierbarkeitsvoraussetzungen behoben werden.

Es droht eine erhebliche Rechtsunsicherheit, wenn Rechtsfolgen an die objektive Einordnung eines Patentes in die Kategorie der "Software-Patente" geknüpft wären. Andererseits greift eine Beschränkung der Kategorie der "Software-Patente" auf solche Patentansprüche zu kurz, die explizit einen Hinweis auf einen programmierten Universalrechner aufweisen. Durch eine solche Beschränkung würden illegitimerweise alle auf den Software-Markt wirkenden Patente aus der Betrachtung ausgeblendet.

Die derzeitige Formulierung des Ausschlusses der «*Datenverarbeitungsprogramme als solche*» vom Patentschutz ist begrifflich verunglückt. Der Wortlaut des §1 PatG und des Artikels 52 EPÜ *sollte deshalb bereinigt werden*.

Verkürzung der Patentlaufzeit ungeeignet

Da "Software-Patente" anhand der Strukturmerkmale ihrer Patentansprüche nicht befriedigend zu greifen sind, werden auch Versuche als untauglich eingestuft, die Auswirkungen auf den Soft-ware-Markt durch Verkürzung der Patentlaufzeit für diese Art von Schutzrechten beispielsweise auf fünf Jahre zu reduzieren.

Eine derartige Verkürzung der Patentlaufzeit wäre aber selbst dann, wenn solche "Software-Patente" zuverlässig abgrenzbar wären, nicht sinnvoll: Die Laufzeit eines Patentes bemisst sich nach dem Zeitablauf ab dem *Anmeldetag*, nicht ab dem Erteilungstag. Der Zeitbedarf des Patentprüfungsverfahrens geht zu Lasten der Zeitdauer, die dem Patentinhaber für die wirtschaftliche Nutzung seines Patentes zur Verfügung stehen. Die Erfahrung zeigt, dass allein die Durchführung des Patentprüfungsverfahrens oft viele Jahre in Anspruch nimmt, so dass viele Patente schon vor der Erteilung durch Zeitablauf verfallen würden. Diejenigen Patente, mit denen Einfluss auf den Software-Markt ausgeübt werden kann, könnten gegebenenfalls jedoch auch in der verbleibenden kurzen Zeit ein patentverletzendes "Open Source"-Produkt vom Markt drängen.

Die Lösung: Privilegierung des Quelltextes

Die zur Absicherung des "Open Source"-Modells gebotene patentrechtliche Abgrenzung der "gewöhnlichen" Patente von "Software-Patenten" sollte daher *nicht an der Art der Patentansprüche*, sondern an der *Art der Verletzungsform* ansetzen. Unsere Empfehlung **PP-1** sieht daher vor, den Umgang mit dem Quelltext von Software patentrechtlich zu privilegieren: Das *Herstellen*, *Anbieten*, *In-Verkehr-bringen*, *Besitzen* oder *Einführen* von Software im Quellcode soll vom Patentschutz

ausgenommen werden. Diese neu einzuführende Privilegierung erstreckt sich jedoch nicht auf den gewerblichen Gebrauch des Programmes durch Ausführung auf einem Computer.

Insbesondere für *Entwickler* von "Open Source"-Software würde durch diese Privilegierung das Risiko einer Patentverletzung vermieden. Wer als *Anwender* eines Datenverarbeitungsprogrammes von einem der bisherigen Privilegierungstatbestände des Patentrechtes begünstigt wird (beispielsweise beim privaten Gebrauch für nichtgewerbliche Zwecke), darf die von den privilegierten Entwicklern und Distributoren bereitgestellte Software ohne patentrechtliche Einschränkungen nutzen. Wer *gewerblich nutzt*, fällt unter das Patent und bedarf für den Gebrauch der Software der Zustimmung des Patentinhabers.

Diese Regelung bietet außerdem einen Anreiz für Software-Firmen, den Quelltext offenzulegen, um zumindest für Entwicklung und Distribution in den Genuss der Privilegierung zu kommen. Dafür ist es nicht erforderlich, den Quelltext urheberrechtlich unter einer bestimmten Lizenz zu lizenzieren. Die Erfahrung zeigt aber, dass auch die Offenlegung des Quelltextes unter einer restriktiveren Lizenz der Einstieg in eine spätere liberale Lizenzierung sein kann.

Neuheitsschonfrist muss (wieder) eingeführt werden

Derzeit gilt im deutschen und EPÜ-Patentrecht auch dasjenige als nicht mehr neu, das vom Erfinder selbst vor der Patentanmeldung verlautbart worden ist.

Will ein Erfinder in den Genuss eines Patentschutzes für seine Erfindung kommen, ist er gezwungen, diese bis zur Einreichung der Patentanmeldung beim Patentamt gegenüber der Öffentlichkeit geheimzuhalten. Eine derartige Geheimhaltungspflicht ist mit Ethos und praktischer Arbeitsweise vieler "Open Source"-Software-Entwickler nicht vereinbar. Der Prozess der Software-Entwicklung wird oft in − im Prinzip für jeden offenen − sich über das Internet konstituierenden Arbeitsgruppen geleistet. Bei der derzeitigen Patentrechtssituation (mit Ausnahme der Vereinigten Staaten) wird so jede Patentierbarkeit bei "Open Source"-Software *in statu nascendi* verhindert.

Wenn der Aufbau eines eigenen Patentpools politisch für vorteilhaft gehalten wird, muss den Entwicklern im "Open Source"-Bereich die Möglichkeit gegeben werden, ihre Erfindungen in patentverwertbarem Zustand zu behalten − ohne die Offenheit ihres Arbeitsmodus aufgeben zu müssen. Daher sollte eine Neuheitsschonfrist von 12 Monaten Dauer, innerhalb der eine Veröffentlichung einer Erfindung durch den Erfinder im entsprechenden Patenterteilungsverfahren nicht als Beitrag zum Stand der Technik zählt, eingeführt werden.

In der Bundesrepublik Deutschland hat es bereits früher eine derartige Neuheitsschonfrist gegeben. Im Hinblick auf das EPÜ dagegen würde es sich nicht um eine Wiedereinführung, sondern um eine Neueinführung handeln. Die Bundesregierung sollte sich dieses Themas nicht nur im Rahmen der nationalen Gesetzgebung, sondern auch im Zusammenhang des "Second Basket" der Diplomatischen Konferenz zur Revision des EPÜ annehmen.

7.2. Kollektive Lizenzierung und Rechtewahrnehmung im Patentbereich

Rechtelizenzierung in Massenverfahren

Im Bereich der Wahrnehmung von Urheberrechten hat man seit langem erkannt, dass die an sich zustimmungspflichtigen Handlungen wie Vervielfältigung oder Verbreitung von Werkskopien in bestimmten Lebensbereichen, etwa im privaten Umfeld, nicht oder nur unter Inkaufnahme einer unverhältnismäßigen Gesetzesdurchsetzung kontrollierbar wären.

Zur Lösung hat man Verwertungsgesellschaften als Instrumente zur kollektiven Lizenzierung von Urheberrechten geschaffen. Die Verwertungsgesellschaften führen eine pauschalisierte Art der Vergütungsabwicklung durch und entlasten sowohl die Rechteinhaber als auch die Rechtenutzer davon, über jeden einzelnen Nutzungsvorgang einen individuellen Vertrag abschließen zu müssen.

Im Patentrecht gibt es bislang keine Parallele.

In vergangenen Zeiten hat sich im Patentbereich keine vergleichbare Problematik gestellt. Die Patentlizenzierung war immer mit der Herstellung oder dem Import einzelner und in der Gesamtzahl beschränkter, rein praktisch nicht nach Belieben vermehrbarer, körperlicher Produkte verknüpft. Durch die moderne Informationstechnologie ist es nunmehr möglich geworden, einen Universalrechner durch ein Vervielfältigungsstück eines Datenverarbeitungsprogrammes in eine "Vorrichtung" zu transformieren, die diesen Patentanspruch verletzt. Die Entscheidung darüber, ob eine Patentlizenz erforderlich ist, hängt also hier häufig nicht mehr von den Eigenschaften körperlicher Produkte, sondern von den Eigenschaften einer Software ab. Software kann ohne nennenswerte Kosten − und im Fall von "Open Source"-Software auch rechtmäßig – in nahezu unbegrenzter Stückzahl vervielfältigt werden.

Damit holt die Problematik der Rechtelizenzierung in ausgesprochenen Massenverfahren, die vom Urheberrecht seit langem bekannt ist, auch das Patentrecht ein.

Lizenzierungskosten sind Transaktionskosten!

Ein besonders vorteilhafter Aspekt beim Einsatz von "Open Source"-Software ist die Unentgeltlichkeit der urheberrechtlichen Lizenzierung. Dies gilt nicht nur im Hinblick auf die absolute Kostenersparnis. Ein ebenso wichtiger Punkt ist, dass kein urheberrechtliches Lizenzverwaltungssystem erforderlich ist. Wenn "Open Source"-Software auf einem weiteren Rechner neu installiert werden soll, so kann dies ohne verwaltungstechnische Abläufe zur Lizenzabrechnung erfolgen. Wird aber durch den Einsatz dieses Datenverarbeitungsprogrammes auf einem Rechner von einer durch einen Patentanspruch geschützten Lehre Gebrauch gemacht, tritt im Fall einer Lizenzierung durch den Patentinhaber das Problem der Lizenzverwaltung erneut auf.

Im Hinblick auf die "Open Source"-Software sind möglicherweise folgende Aspekte von Bedeutung:

- Auf freiwilliger Basis könnte Patentinhabern – auch solchen außerhalb der "Open Source"- Szene − der Beitritt zu einer Verwertungsinstitution für Patentrechte eröffnet werden.

Diese Patentinhaber treten die Rechte aus dem Patent an die Institution ab und erhalten einen Anteil an deren Gewinnausschüttung. Diese Verwertungsinstitution schließt dann mit interessierten Lizenznehmern, z. B. Unternehmen, Pauschalverträge über alle in ihrem Portfolio befindlichen Patente ab. Die Bemessungsgrundlage für die Pauschalverträge ist nicht jede einzelne Nutzung des Patentes. Es wird vielmehr auf leichter handhabbare Parameter wie z.B. die Betriebsgröße abgestellt.

- Falls Patentinhabern eine Art von "Zwangslizenzierung" ihrer Schutzrechte (ähnlich wie bei Urheberrecht, Rundfunkprivileg etc.) auferlegt werden sollte, könnte eine solche Institution ein kompensatorisches Entgelt einsammeln und an die Rechteinhaber verteilen.
- Weitere Aufgaben für solche Institutionen wären denkbar.

  Es wäre vorstellbar, dass einzelne Erfinder ihre Rechte vor dem Einreichen einer Patentanmeldung an die Verwertungsinstitution abtreten und dafür entweder finanziell und/oder durch unentgeltliche Lizenzierung (aus dem Lizenzpool) belohnt werden. Ein eigener Patentpool könnte auch dazu benutzt werden, Patentinhaber außerhalb der "Open Source"-Community zum Verzicht auf die Durchsetzung patentrechtlicher Unterlassungsansprüche zu gewinnen. Sie könnten sich stattdessen an der kollektiven Lizenzierung beteiligen.

Besonders für die letzte Variante ist am Anfang ein erheblicher Finanzierungsbedarf erforderlich. Wenn der Öffentlichen Hand aus übergeordneten wirtschaftspolitischen Motiven heraus daran gelegen ist, den Einsatz von "Open Source"-Software zu fördern, könnte eine Anschubfinanzierung aus Steuermitteln eine sinnvolle Subventionierung darstellen.

Keine Zwangsabgaben

Insbesondere aus kartellrechtlichen Gründen scheint es kaum vorstellbar, im vorhandenen Gesetzesrahmen auf rein privatrechtlicher Grundlage eine derartige Verwertungsinstitution zu begründen.

Es wäre sinnvoll, eine dem Urheberrechtswahrnehmungsgesetz vergleichbare Rechtsgrundlage zu schaffen. Im Gegensatz zur urheberrechtlichen Situation, bei der auch Zwangsabgaben (Geräteabgaben) vorgesehen sind, sollte es hinsichtlich des Patentwesens so weitgehend wie möglich bei einem System der Freiwilligkeit belassen werden. Es sollte Patentinhabern und Patentnutzern grundsätzlich freigestellt sein, ob sie sich der Verwertungsinstitution anschließen. In den Genuss der − konfliktvermeidenden − Kollektivlizenzierung kommen dann eben nur diejenigen Patentinhaber und -nutzer, die sich freiwillig dazu entschlossen haben. Zwischen Einzellizenzierung oder Durchsetzung des patentrechtlichen Unterlassungsanspruches einerseits und der Verwertungsinstitution auf der anderern Seite könnte eine Konkurrenzsituation entstehen, die zu einem lebendigen Markt mit neuen Lizenzierungsmodellen führt.

7.3. Interoperabilität

Ein gewichtiges Problem stellt die *Förderung der Interoperabilität* zwischen unterschiedlichen IT--Plattformen dar.

Im Urheberrecht ist durch §69e UrhG das generelle *Dekompilationsverbot* aus §69c Nr. 2 UrhG in bestimmten Fällen durchbrochen worden, die der Herstellung von Interoperabilität dienen.

Interoperabilität wird von der EG-Computerrechtsrichtlinie als die Fähigkeit eines Programmes zum Austausch von Informationen und zur wechselseitigen Verwendung der ausgetauschten Informationen definiert.[264] Die vom Gesetzgeber urheberrechtlich privilegierte Herstellung von Interoperabilität kann jedoch auch patentrechtlich vereitelt werden, wodurch erhebliche Marktzutrittsbarrieren geschaffen werden können, deren ökonomischer Wirkungsradius weit über den eigentlichen Patentschutzgegenstand hinausgeht. Es stellt sich die Frage, ob hier der Grundansatz der Zwangslizenzierung aus §24 Abs. 1 PatG im Rahmen einer zu schaffenden gesetzgeberischen Lösung ein problemadäquates Instrumentarium bieten könnte.

Die Anwendung des bestehenden §24 Abs. 1 PatG ist nur bei *nachgewiesenem öffentlichem Interesse* möglich. Eine Abwägung der entgegenstehenden Interessen von Patentinhaber und an der Herstellung von Interoperabilität interessiertem Software-Entwickler oder -Anwender als Grundlage für einen eventuellen gesetzgeberischen Eingriff setzt ein makroökonomisches Verständnis der Strukturen in entwickelten Informationsgesellschaften voraus − welches heute noch nicht zur Verfügung steht. Vor tiefgreifenden rechtspolitischen Maßnahmen sollte eine gründliche Erforschung der ökonomischen Implikationen erfolgen.

Wir geben zu bedenken, dass sich Zwangslizenzen im Urheberrecht als ein probates Mittel zur Auflösung von ähnlichen Zielkonflikten bewährt haben.

7.4. Interaktionen zwischen Patent- und Urheberrecht

In einer Zeit, in der mit Software zumindest eine mittelbare Patentverletzung begangen werden kann, muss es als Anachronismus erscheinen, wenn derjenige, der den Quelltext nicht offenlegt, mit einer zweifachen patentrechtlichen Privilegierung belohnt wird: Zum einen gelten die in dem Code verwirklichten Ideen und Grundsätze als der Öffentlichkeit nicht zugänglich. Eine Patentierung dieser Ideen und Grundsätze ist auch dann noch möglich, wenn die Software − als Binärcode − bereits allgemein erhältlich ist. Zum anderen muss derjenige, der durch Dekompilation eine Patentverletzung nachzuweisen sucht, mit umgehenden Gegenansprüchen aus dem urheberrechtlichen Dekompilationsverbot heraus rechnen.

7.5. Unterrichtung der Öffentlichkeit

Die Patentämter sind gesetzlich dazu zu verpflichten, offengelegte Patentschriften, erteilte Patente und Rechtsstandsdaten für alle noch anhängigen Anmeldungen bzw. noch "lebenden" Patente vollständig und unentgeltlich im Internet zum Abruf durch die Öffentlichkeit bereitzuhalten.

Ein in seiner Art vorbildliches Patent- und Schutzmarkeninformationssystem für die Öffentlichkeit auf Basis des Internet betreibt das U.S.-Patentamt. Über das Internet sind dort alle Patentinformationen *kostenlos* einsehbar. Diesem Beispiel sollte auch die Bundesrepublik Deutschland folgen.

---

264 **Fromm/Nordemann 1998,** §69e Rdn. 1, S. 495.

Für die breite Öffentlichkeit im Allgemeinen und für die Entwickler von "Open Source"-Software im Besonderen muss der Zugang zu Patentinformationen erleichtert werden. Derzeit sieht die Bundesregierung Patentinformationen als "geldwertes Gut" an, welches an den Bürger nur gegen Entgelt abgegeben werden darf. Beispielsweise kostet ein Abruf eines Patentrechtsstandsberichtes DEM 4,00.[265]

Abgesehen davon, dass der Patentinhaber bereits durch die an das Patentamt abgeführten Anmeldegebühren für die Veröffentlichung des erteilten Patentes ein Entgelt entrichtet hat, sollte der negative Effekt beachtet werden, der durch die derzeitige Intransparenz für den Bürger entsteht. Zwar beteiligt sich die Bundesrepublik Deutschland an dem vom Europäischen Patentamt koordinierten Patentdatenbank-Projekt ***esp@acenet***. Bei genauerer Betrachtung erweist sich der deutsche Beitrag zu diesem System jedoch als eher symbolisch: Deutsche *Patentanmeldungen* sind auf die letzten zwei Jahre beschränkt und *erteilte deutsche Patente* werden überhaupt nicht berücksichtigt.

7.6. Internationales Recht

Bei den anstehenden Beratungen im Rahmen des "WIPO Standing Committee on the Law of Patents (SCP)" bietet sich der Bundesregierung eine gute Gelegenheit, sich im Rahmen der dort angesetzten Verhandlungen dafür einzusetzen, eine internationale Einführung des in Empfehlung **PP-1** angesprochenen Quellcodeprivilegs und eine Harmonisierung des einschlägigen Internationalen Privatrechtes vorzunehmen.

Durch Beratungen auf der Ebene der WIPO wäre eine Lösung dieser Probleme im Prinzip erzielbar.

---

265 Vgl. **Rötzer 1999.**

8. Ergebnisse und Ausblick

8.1. Der im November 2000 erreichte Diskussionsstand

Die Diplomatische Konferenz zur Revision des Europäischen Patentübereinkommens (EPÜ) hat eine Änderung des Wortlauts von Art. 52 EPÜ zunächst abgelehnt. Die deutsche und europäische Politik hat daher Zeit gewonnen, das Problem der Patentierbarkeit von Software nochmals zu durchdenken.

Natürlich konnte dieses Kurzgutachten nur vorläufige Ergebnisse zu Tage fördern. Manche unserer Ergebnisse scheinen so plausibel, dass wir schon jetzt empfehlen, die Patentpolitik – mit Bedacht allerdings – zu verändern.

Die deutsche und angelsächsische Literatur hat sich dem Thema "Software-Patente" in den vergangenen Jahrzehnten mit einiger Sorgfalt gewidmet. Dabei gab es eine zunehmende Divergenz zwischen dem "*Law on the books*" und dem "*Law in action*": Unbeschadet des Wortlauts des deutschen und amerikanischen Patentrechts haben sich Gerichte in Europa und den USA nicht gescheut, "Software-Patente" rechtlich anzuerkennen. Das *Ob* von "Software-Patenten" ist nicht mehr ernsthaft bestritten.

Dieser Wandel hat sich in gut 30 Jahren vollzogen. Noch 1968 konnte die New York Times die damalige U.S.-amerikanische Rechtsauffassung, der Europa mit Verspätung noch immer gefolgt ist, so zusammenfassen:

> «Software is unpatentable»[266]

Zu Beginn des Jahres 2000 unterrichtete dieselbe Zeitung ihre Leser und Leserinnen mit einer entgegengesetzten Botschaft:

> «Patently absurd. Once the province of a nuts-and-bolts world, patents are now being applied to thoughts and ideas in cyberspace. It‘s a ridiculous phenomenon, and it could kill e-commerce.»[267]

---

266 **Jones 1968**.

267 **Gleick 2000**.

Es muss in den letzten dreißig Jahren gewichtige gesellschaftliche und ökonomische Veränderungen gegeben haben, um solche diametralen Bewertungen desselben Sachverhalts erklären zu können. Aber welche?

Die in der Wissenschaft noch überwiegende Meinung tut sich mit Erklärungen schwer:

Die in Deutschland unter Juristen vorherrschende Sicht des Themas übersieht häufig den ökonomischen Kern von Patenten. Entsprechend bleiben die Argumente für oder gegen "Software-Patente" zumeist beliebig.

Die in Deutschland unter Mikroökonomen vorherrschende Meinung hat überwiegend eine klassische Sicht auf das Patentthema. Sie sieht das Patent als Waffe im Wettbewerb, als gesetzlich geschützten Marktvorteil. Sie kann beispielsweise nicht erklären, weshalb viele Unternehmen – große wie kleine und mittlere, Entwickler und Anwender von Software – zunehmend "Open Source"-Software entwickeln bzw. einsetzen, die gegenwärtig *gerade nicht* unter das Patentparadigma fällt.

In einer so unklaren Situation wäre eine Unterstützung durch makroökonomische Modelle hilfreich. Aktuelle Modelle für Deutschland sind jedoch nicht bekannt. In der angeheizten Diskussion im Vorfeld der Münchner Konferenz vom November 2000 fehlte in kaum einer Stellungnahme ein Hinweis auf das Modell der MIT-Ökonomen *Bessen* und *Maskin*.[268] Aus diesem Modell lässt sich aber höchstens eine – aus unserer Sicht plausible – These generieren. Überspitzt könnte sie lauten:

> Die Innovationsgeschwindigkeit auf den Software-Märkten und in den Unternehmen ist umso größer, je weniger sie durch Patente und andere Monopole behindert wird.

Gesichert ist das von den Autoren präsentierte Wissen jedoch nicht − die wissenschaftliche Ausgangslage ist noch unbefriedigend. Es lohnt sich also, daran weiterzuarbeiten.

8.2. Die Ursachen des unbefriedigenden Diskussionsstandes

Unsere Untersuchungen lassen es plausibel erscheinen, dass für diesen Zustand drei, hier interessierende, Ursachen maßgeblich sein könnten:

- Der wachsende E-Commerce und die stetig steigende Bedeutung des Internet haben die Innovationsgeschwindigkeit in den Unternehmen dramatisch beschleunigt.
- Die Unternehmen reagieren auf diesen Sachverhalt mit neuen Innovationsstrategien:

  Unitechnologische Innovationen werden durch multitechnologische substituiert. Wichtiger Bestandteil dieser neuen Strategie ist die Integration von "Open Source"-Software in die

---

268 **Bessen/Maskin 2000**.

Unternehmensnetzwerke.[269]

- "Open Source"-Software ist ein neues − marktwirtschaftliches − Modell der Software-Entwicklung und des Software-Vertriebs, das sich mit dem jetzt auf der Welt geltenden Patentrecht nicht ohne Weiteres in Einklang bringen lässt. Dieses neue Entwicklungsmodell ist Entwicklungsmodellen aus der Vergangenheit wahrscheinlich überlegen.

Die Debatte um "Software-Patente" musste also bisher im Ergebnis unbefriedigend bleiben: Die einen übersehen den ökonomischen Kern von Patenten und haben für den Gedanken von "Open Source"-Software in der Regel nur Unverständnis übrig. Die anderen bleiben veralteten Innovationsstrategien verhaftet. Selbst wenn Patente in der Vergangenheit Software-Innovationen gefördert haben sollten, muss das nicht heißen, dass sie dies auch in der Zukunft tun.

Das Gegenteil könnte der Fall sein.

8.3. Was künftig zu tun ist

In diesem Kurzgutachten ist – so hoffen wir – die folgende These plausibel geworden:

> Über die Zukunft von computer-implementierten bzw. computer-implementierbaren Erfindungen kann man rational nur entscheiden, wenn die Besonderheiten von "Open Source"-Software berücksichtigt werden. "Open Source"-Software ist:
>
> - ein völlig neuartiges Modell der Software-Entwicklung
> - ein völlig neuartiges Modell des Software-Vertriebs

In parallelen Arbeiten haben die "Forschungsgruppe Internet Governance" an der TU-Berlin[270] und eine Forschungsgruppe am Institut für Wirtschaftsinformatik in Saarbrücken[271] unabhängig voneinander zeigen können, dass man die beschriebenen Lücken in der wissenschaftlichen Durchdringung des Themas überwinden kann, wenn man mit einer veränderten Fragestellung an die mikroökonomische Perspektive herangeht. Die Frage ist nicht mehr: Wie können kleine und mittlere Unternehmen der "Open Source"-Software-Branche vor übermächtigen Wettbewerbern geschützt werden? Die Frage ist vielmehr:

---

269 Vgl. dazu **Nüttgens/Tensei 2000 c.**

270 **Ardal 2000**.

271 **Nüttgens/Tensei 2000 a**; **Nüttgens/Tensei 2000 b**; **Nüttgens/Tensei 2000 c**.

Welche Eigenschaften machen "Open Source"-Software-Produkte (häufiger) im Wettbewerb überlegen?

Die Arbeiten gleichen sich im Ergebnis, beide Forschungsgruppen schlagen aber etwas andere Richtungen bei der Bearbeitung des Themas ein – Berlin sieht das Problem eher aus der Sicht der Informatik und der Rechtswissenschaft, Saarbrücken eher aus der Sicht der Mikroökonomie. An den erwähnten Arbeiten waren sehr junge Wissenschaftler beteiligt.

Das Verständnis dieses neuen Modells der Software-Entwicklung und des -Vertriebs muss im Mittelpunkt der künftigen Patentpolitik stehen. Wir schlagen hierzu also eine Verbindung von mikroökonomischer und informatischer Betrachtung vor.

Daraus leitet sich ein weiterer Schwerpunkt ab: Die Politik muss für ihre Entscheidungen wissen, inwieweit es sich bei "Open Source"-Software um ein quantitativ gewichtiges Problem handelt, wie sich die behaupteten Effekte verteilen, womit man bei den einzelnen Unternehmenstypen zu rechnen hat. Man muss die Antriebe, "Open Source"-Software in Unternehmensnetzwerke einzubauen, sehr genau kennen.

Aus der Literatur und Kontakten zu Unternehmen sind wir uns über die folgende **Aussage** sicher:

> Die Wettbewerbsfähigkeit der Unternehmen nimmt zu, in dem Maße, wie sie quelloffene Software in ihr Portfolio einbeziehen.

Von der Offenheit der Sourcen profitieren sowohl kleine und mittlere als auch größere Unternehmen.

Es sollte empirisch genau untersucht werden, ob der behauptete Zusammenhang (auch) allgemein zutrifft. Drei Schwerpunkte der empirischen Forschung sind vordringlich:

1. **Das Innovationsverhalten kleiner und mittlerer Unternehmen** (jeweils unterschieden nach Anwendung und Entwicklung von Software)

   Die Software-Entwicklung vollzieht sich in Deutschland und der Europäischen Union überwiegend in kleinen und mittleren Unternehmen.

2. **Das Verhalten von Großunternehmen**

   Hierbei muss man sowohl an Unternehmen wie IBM als auch SAP denken. Völlig unabhängig von der Frage, wie man zu "Software-Patenten" steht, haben gerade Großunternehmen ein weiteres Problem mit Patenten: Die innere Hierarchie eines Großunternehmens wird maßgeblich dadurch gebildet, dass die Mitarbeiter sich ihren betrieblichen Rang über Patente erarbeiten. Es ist ungeklärt, wie sich das Unternehmen strukturieren müsste, wenn der Patentschutz ganz oder teilweise nicht (mehr) gewährleistet ist.

3. **Auswertung der Patentregister in Stichproben**

   Wir haben in diesem Gutachten ausgeführt, dass es weltweit keine verlässlichen Zahlen über die erteilten "Software-Patente" gibt. Der Begriff ist zu unscharf und lässt deshalb

keine klare Abgrenzung zu "Nicht-Software-Patenten" zu. Die entsprechenden Register sind bisher noch nirgendwo ausgewertet worden. Eine zunächst stichprobenartige Auswertung sollte weitergehende Erkenntnisse liefern.

Damit allein kann das Innovationsgeschehen im Bereich der Software-Entwicklung und des Software-Vertriebs noch nicht zulänglich erklärt werden. Seit kurzer Zeit sind Zahlen bekannt, die darauf hindeuten, dass es einen spezifisch deutschen Beitrag in diesem Bereich gibt. Nach Zahlen einer empirischen Untersuchung von 1999 gibt es weltweit ca. 250.000 "Open Source"-Entwickler.[272] Demnach sind europäische Entwickler weltweit dominierend, deutsche Entwickler belegen die zweite Stelle. Die Autoren der Studie werden zitiert mit den Worten:

> «We knew the Germans were really active, but we didn't know how active. We were really knocked out when we saw the Germans were the second largest contributors.»[273]

Weshalb gerade deutsche Entwickler im "Open Source"-Bereich so stark engagiert sind, lässt sich mit bekannten Argumenten nicht erklären. Man muss deshalb vermuten, dass es einen spezifisch deutschen Beitrag im Bereich der Software-Innovationen gibt. Soweit wir sehen, ist diese Vermutung noch nicht untersucht worden.[274]

Die drei Schwerpunkte – Modellbildung, Überprüfung des Modells, Untersuchung des spezifisch deutschen Beitrages – bilden den Hintergrund, vor dem die Bundesregierung die künftige Patentpolitik gestalten sollte.

Die Bundesregierung sollte sich **drei Ziele** setzen:

Das Wissen um die Innovationsdynamik moderner Software-Entwicklung muss entschieden verbreitert werden.

---

272 **Demsey/Weiss/Jones/Greenberg 1999**.

273 Zit. nach **Glascock 2000**, der erstmals über diese Studie berichtete.

274 Diesen Zusammenhang übersieht die Studie **Hart/Holmes/Reid 2000**, die die Europäische Kommission noch im Vorfeld der Münchner Konferenz veröffentlicht hat. Die Fragestellung hätte lauten müssen: Ist "Open Source"-Software ein eigenständiger europäischer Beitrag für den Bereich von Software-Innovationen und könnte gerade dieser Beitrag eine überlegene Wettbewerbsposition der Europäischen Union gegenüber den USA begründen? Deuten − mit anderen Worten − die Zahlen auf eine Entwicklung hin, wie sie für den Mobilfunk typisch ist? Weil die Autoren diese Frage nicht gestellt haben, musste ihnen die hier angeschnittene Innovationsdynamik in den Unternehmen entgehen. Entsprechend bleiben ihre Aussagen zur Patentpolitik beliebig.

Der hier entfalteten Vorschläge sollten dazu beitragen.

Man kann gewünschte Entwicklungen im Bereich der Software-Innovationen fördern, ohne das System des Patentrechts über Bord zu werfen.

Dieses Kurzgutachten enthält eine Vielzahl von Vorschlägen, die einen Interessenausgleich besser ermöglichen als die geltenden Regularien. Nicht das Ob von "Software-Patenten" muss im Zentrum der Überlegungen stehen, das Wie steht zur Debatte. Der "Berliner Vorschlag zu 'Open Software Patents'"[275] ist eine Anregung für die anstehende Diskussion.

Man muss die europäische Perspektive einer künftigen Patentpolitik präziser definieren.

Das Gutachten hat hierfür noch keine Antwort geliefert. Es wird aber eine aussichtsreiche Fragestellung formuliert.

Man braucht mehr Wissen in den von uns skizzierten Bereichen. Man braucht aber auch deutlich mehr Wissen über das Zusammenspiel von Urheberrecht und Patentrecht. Das Verfassungsrecht ist ebenfalls gefragt: Wenn wirklich «*thoughts and ideas*» durch das überkommene System des Gewerblichen Rechtsschutzes geschützt sein sollten, wie die New York Times in dem eingangs zitierten Aufsatz behauptet[276], wird die Abgrenzung zum Recht der Öffentlichkeit auf Informationsfreiheit zunehmend problematisch. Man wird sich dann daran erinnern müssen, dass die Bipolarität des Schutzsystems ein Kind des 19. Jahrhunderts ist. Vieles spricht dafür, dass eine alte ökonomische Frage erneut gestellt werden muss:

«Patents and Copyrights: Do the benefits exceed the costs?»[277]

---

275 **Gehring 2000**.

276 **Jones 1968**.

277 So der Titel den Aufsatzes von **Cole 2000**, der auf der letzten Jahrestagung der Mont Pèlerin Society Ende November 2000 in Santiago lebhaft und kontrovers diskutiert worden ist; siehe dazu den Bericht von **Schwarz 2000**.

Es fehlt sicher nicht an Stimmen, die die vorhandenen Schutzsysteme verdammen oder verteidigen.

Es fehlt jedoch an empirisch belegtem Wissen und an Einsicht in das, was der Fall ist. Dieses Gutachten versteht sich als Beitrag zur dringend gebotenen Schließung dieser Lücken.

9. Literaturverzeichnis

Altes/Dommering/Hugenholtz/Kabel 1992
Willem F. Altes, Egbert J. Dommering, P. Bernt Hugenholtz, Jan J. C. Kabel: **Information Law Towards the 21st Century.** Kluwer, Deventer & Boston 1992.

Anderson 1993
Ross Anderson: **Why Cryptosystems Fail.** ACM 1st Conference Computer and Communication Security '93, im Internet: http://www.attrition.org/~wrlwnd/crypto/cryptanalysis/wcf.html (18.10.2000).

Anderson 1997
Ross Anderson: **GSM hack − operator flunks the challenge.** In: The Risks Digest, Forum on Risks to the Public in Computers and Related Systems, Volume 19, Issue 48, Friday, 5 December 1997, im Internet: http://catless.ncl.ac.uk/Risks/19.48.html (3.12.2000).

Ardal 2000:
Atila Ardal: **Open Source − das Beispiel Linux. Ökonomische Analyse und Entwicklungsmodell eines erfolgreichen Betriebssystems.** Diplomarbeit am Fachbereich Informatik der Technischen Universität Berlin, Mai 2000, im Internet: http://ig.cs.tu-berlin.de/da/059/index.html (30.10.2000).

Banse/Langenbach 1999
Gerhard Banse, Christian J. Langenbach: **Geistiges Eigentum und Copyright im multimedialen Zeitalter. Positionen, Probleme, Perspektiven.** Hg. von der Europäischen Akademie Bad Neuenahr-Ahrweiler. 2.Aufl., Nr. 13 der «Grauen Reihe». Bad Neuenahr-Ahrweiler 1999.

Barbrook 2000
Richard Barbrook: **Die Regulierung der Freiheit.** In: Telepolis Mix v. 14.9.2000, im Internet: http://www.heise.de/tp/deutsch/special/med/8725/1.html (30.10.2000).

Bartsch 2000
Michael Bartsch: **Computerviren und Produkthaftung.** In: CR 11/2000, S. 721-725.

Bechtold 2000
Stefan Bechthold: **Software Patents in Germany - Current Developments.** In: Cyberspace Law Abstracts No. 53: November 20, 2000, im Internet: http://papers2.ssrn.com/paper.taf?abstract_id=240205 (22.11.2000).

Behrens 2000
Christian-Uwe Behrens: **Makroökonomie. Wirtschaftspolitik.** Oldenbourg Verlag, München, Wien 2000.

Benjamin/Blunt 1993
Robert I. Benjamin, Jon Blunt: **Informationstechnik im Jahr 2000 − Ein Wegweiser für Manager.** In: Harvard Business manager: Informations- und Datentechnik, Band 2, manager magazin Verlagsgesellschaft, Hamburg (ohne Jahresangabe).

Benkard 1993
Georg Benkard et al.: Beck'sche Kurzkommentare, Band 4: **Patentgesetz, Gebrauchsmustergesetz.** 9. Aufl., C.H. Beck, München 1993.

Bericht zur Software-Richtlinie (91/250/EWG) 2000
Bericht der Kommission an den Rat, das Europäische Parlament und den Wirtschafts- und Sozialausschuss über die Umsetzung und die Auswirkungen der Richtlinie 91/250/EWG über den Rechtsschutz von Computerprogrammen vom 10. April 2000. Im Internet: http://europa.eu.int/comm/internal_market/en/intprop/intprop/docs/reportde.pdf (2.12.2000).

Berman 2000
Howard L. Berman: **Introduction of H.R. 5364, the Business Method Patent Improvement Act of 2000.** Rede vor dem Kongress in Washington D. C. am 3.10.2000, im Internet: http://www.house.gov/berman/floor100300.htm (3.12.2000).

Bessen/Maskin 2000
James Bessen, Eric Maskin: **Sequential Innovation, Patents, and Imitation.** Working Paper, Department of Economics/Massachusetts Institute of Technology No. 00-01, January 2000, Cambridge 2000, im Internet: http://www.researchoninnovation.org/patent.pdf (1.10.2000).

Besen 1998
Stanley M. Besen: **Intellectual Property.** In: Palgrave Dictionary of Economics & Law. Vol. 2, p. 348ff.

Betten 1995
Jürgen Betten: **Patentschutz von Computerprogrammen.** In: GRUR 1995, S. 775ff.

Bömer 1989
Roland Bömer: **Risikozuweisung für unvermeidbare Softwarefehler.** In: CR 1989, S. 361-367.

Born 2000
Achim Born: **Keine Entspannung. CDI-Stellenmarkt-Analyse 2000.** In: iX 12/2000, S. 46f.

Boyes/Melvin 1999
William Boyes, Michael Melvin: **Economics.** 4th ed., Houghton Mifflin Company, Boston, New York 1999.

Bradner 1999
Scott Bradner: **The Internet Engineering Task Force.** In: DiBona/Ockman/Stone 1999, pp 47-52.

BSI DDoS 2000
Bundesam für Sicherheit in der Informationstechnik: **Empfehlungen zum Schutz vor verteilten Denial of Service-Angriffen im Internet.** Version 1.1 vom 20.6.2000, im Internet: http://www.bsi.bund.de/ddos.html (3.12.2000).

Busche 2000
Jan Busche: **Der Schutz von Computerprogrammen − Eine Ordnungsaufgabe für Urheber- und Patentrecht?** In: Mitteilungen der deutschen Patentanwälte 2000, S. 164ff.

Buschmann et.al. 1996
Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: **Pattern-Oriented Software Architecture. A System of Patterns.** John Wiley & Sons, 1996.

Busse 1999
Rudolf Busse: **Patentgesetz. Kommentar.** 5.Aufl., Grundlagen des deutschen Patentrechts, De Gruyter, Berlin 1999.

Chamberlin 1998
Donald D. Chamberlin: **Sharing our Planet.** In: Peter J. Denning, Robert M. Metcalfe (ed.): **Beyond Calculation. The Next Fifty Years of Computing.** Copernicus (Springer), New York 1998, S. 233-243.

Cohen 1999
Seth A. Cohen: **To Innovate or Not to Innovate, That is the Question: The Functions, Failures, and Foibles of the Reward Function.** 5 Michigan Telecommunications Technology Law Review 1 (1999), Im Internet: http://www.mttlr.org/volfive/cohen.pdf (12.10.2000).

Cohen/Lemley 2001
Julie E. Cohen, Mark A. Lemley: **Patent Scope and Innovation in the Software Industry.** Erscheint in California Law Review 89 (2001), January 2001, preprint im Internet: http://www.law.georgetown.edu/faculty/jec/publications.html (6.9.2000).

Cole 2000
Julio H. Cole: **Patents and Copyrights: Do the Benefits Exceed the Costs?** Im Internet: http://www.economia.ufm.edu.gt/Catedraticos/jhcole/Cole%20_MPS_.pdf (26.11.2000).

Conrads 2000
Martin Conrads: **»Der Teufel steckt im Detail« Zur Tagung «Wem gehört das Wissen?** – **Geistiges Eigentum im Zeitalter des Internet».** In: Teleopolis v. 30.10.2000, im Internet: http://www.heise.de/tp/deutsch/inhalt/konf/4150/1.html (3.12.2000).

Cooper et al. 1995
Frederick J. Cooper, Chris Goggans, John K. Halvey, Larry Hughes, Lisa Morgan, Karanjit Siyan, William Stallings, Peter Stepehenson: **Implementing Internet Security.** New Riders Publishing, Indianapolis 1995.

Cooper Dreyfuss/Rosenthal Kwall 1996 (1999)
Rochelle Cooper Dreyfuss, Roberta Rosenthal Kwall: **Intellectual Property: Trademark, Copyright and Patent Law: Cases and materials.** Foundation Press, Westbury, New York 1996, with a supplement from 1999.

Delio 2000
Michelle Delio: **It'll Be an Open-Source World.** Wired News v. 15. August 2000, im Internet: http://www.wired.com/news/technology/0,128238240,00.html (3.12.2000).

de Paoli 2000
Nicola de Paoli: **101 Köpfe der New Economy: Wolfgang Tauchert.** In: Financial Times Deutschland v. 30.10.2000.

Dempsey/Weiss/Jones/Greenberg 1999
Bert J. Dempsey, Debra Weiss, Paul Jones, and Jane Greenberg: **A Quantitative Profile of a Community of Open Source Linux Developers.** UNC Open Source Research Team. School of Information and Library Science University of North Carolina at Chapel Hill. October 6, 1999, im Internet: http://www.ibiblio.org/osrt/develpro.html (11.2000).

DiBona/Ockman/Stone 1999
Chris DiBona, Sam Ockman, Mark Stone (ed.): **Open Sources. Voices from the Open Source Revolution.** O'Reilly & Ass., Sebastopol, CA, 1999.

Digital Dilemma 2000
Computer Science and Telecommunications Board/National Research Council: **The Digital Dilemma. Intellectual Property in the Information Age.** Washington, D.C.: National Academy Press 2000.

Dolfsma 2000
Dolfsma, Wilfried: **How Will the Music Industry Weather the Globalization Storm?** In: FirstMonday Vol. 5 Iss. 5 (May 2000), im Internet: http://firstmonday.org/issues/issue5_5/dolfsma (18.5.2000).

Elkin-Koren/Salzberger 1999
Nina Elkin-Koren, Eli M. Salzberger: **Law and Economics in Cyberspace.** In: International Review of Law and Economics Vol. 19 No. 4 December 1999, p. 553ff.

Ellins 1997
Julia Ellins: **Copyright Law, Urheberrecht und ihre Harmonisierung in der Europäischen gemeinschaft.** Duncker & Humblot, Berlin 1997.

Endres 2000
Albert Endres: **«Open Source» und die Zukunft der Software.** In: Informatik Spektrum vom 23. Oktober 2000, S. 316ff.

Engel 1993
Friedrich-Wilhelm Engel: **Über "Computerprogramme als solche"**. In: GRUR 1993, S. 194ff.

EU Working Group 1999
European Union Working Group on Libre Software: **Free Software/Open Source: Information Society Opportunities for Europe?** December 1999, Version 1.0, p. 22; zit. nach Smarr/Graham 2000, p. 4.

Esslinger/Betten 2000
Alexander Esslinger, Jürgen Betten: **Patentschutz im Internet.** CR 1/2000, S. 18-22.

Esslinger/Hössle 1999
Alexander Esslinger, Markus Hössle: **Zur Entscheidung «State Street v. Signature Financial» des amerikanischen Court of Appeals for the Federal Circuit.** In: Mitteilungen der deutschen Patentanwälte 1999, S. 327ff.

Eucken 1990
Walter Eucken: **Grundsätze der Wirtschaftspolitik.** 6. Aufl., UTB, Tübingen 1990.

Evans/Wurster 2000
Philip Evans, Thomas S. Wurster: **WEB ATT@CK.** Hanser Verlag, München 2000.

Friedman 1998 a
David D. Friedman: **Computer Law.** In: Palgrave Dictionary of Economics & Law. Vol. 1, p. 377ff.

Friedman 1998 b
David D. Friedman: **Trade secret.** In: Palgrave Dictionary of Economics & Law. Vol. 3, p. 604ff.

Fromm/Nordemann 1998
Wilhelm Nordemann, Kai Vinck, Paul W. Hertin: **Urheberrecht.** Kommentar. 9. Aufl., Verlag W. Kohlhammer, Stuttgart 1998.

Gamma et al. 1995
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: **Design Patterns. Elements of Reusable Object-Oriented Software.** Addison-Wesley, Reading, MA, u.a. 1995.

Gehring 1996
Robert Gehring: **Freeware, Shareware und Public Domain.** Studienarbeit am Fachbereich Informatik, TU Berlin, 1996, im Internet: http://www.ig.cs.tu-berlin.de/sa/043/index.html (3.12.2000).

Gehring 2000
Robert Gehring: **Berliner Ansatz zu *"*Open Software Patents*"*. Ein Ausweg aus dem *"Digital Dilemma"*?**, Beitrag zur Konferenz über wirtschaftspolitische Aspekte der Patentierung von Software, veranstaltet vom Bundesministerium für Wirtschaft und Technologie am 18. Mai 2000 in Berlin, im Internet: http://www.sicherheit-im-internet.de/themes/print.phtml?ttid=2&tdi.d=75 (3.12.2000) bzw. http://www.Think-Ahead.ORG/Cyberlaw/ (30.10.2000).

Gengler 1998
Barbara Gengler: **IT-Manager starten Protestaktion gegen neues Softwaregesetz. US-Programmanbieter wollen nicht für ihre Produkte haften.** In: Computer Zeitung 48/1998, S. 2.

Ghidini 1997
Gustavo Ghidini: **"Protektionistische Tendenzen" im gewerblichen Rechtsschutz.** In: GRUR int 1997, S. 773ff.

Ghosh 1999
Rishib A. Ghosh: **Cooking Pot Markets: An Economic Model for the Trade in Free Goods and Services on the Internet.** First Monday Vol. 3 Issue 3, im Internet: http://www.firstmonday.dk/issues/issue3_3/ghosh/index.html (18.10.2000).

Glascock 2000
Stuart Glascock: **Germany Leads In Open-Source Development.** In: TechWeb News. Nov 1, 2000, im Internet http://www.techweb.com/wire/story/TWB20001101S0016 (2.11.2000).

Gleick 2000
James Gleick: **Patently Absurd.** New York Times Magazine v. 12.3.2000.

Goldstein 1999
Paul Goldstein: **Copyright, Patent, Trademark and Related State Doctrines. Cases and materials on the law of intellectual property.** 4th ed, Foundation Press, New York 1999.

Gorny 1986
Peter Gorny: **Kategorien von Softwarefehlern.** In: CR 1986, S. 673-677.

Gotta 2000
Frank Gotta: **Quellcode aus dem Baukasten lockt die Programmierer ins Web.** In: Computer Zeitung v. 9.11.2000, S .13.

Graff 2000
Bernd Graff: **Wer die Quelle hat, hat die Wahl. Das Geheimnis von Microsoft** – **Ein Blick auf den Programmcode ist wie ein Blick in die Seele.** In: Süddeutsche Zeitung v. 30.10.2000.

Gross 2000
Grant Gross: **Patent Office Director: My hands are tied.** In: Newsforge v. 4.10.2000, im Internet: http://www.newsforge.com/article.pl?sid=00/10/04/2035225 (5.10.2000).

Hadfield 2000
Gillian Hadfield: **Privatizing Commercial Law: Lessons from the Middle and the Digital Ages.** Social Science Research Network Electronic Paper Collection of April 21, 2000, http://papers.ssrn.com/paper.taf?abstract_id=220252 (15.8.2000).

Handbuch des EU-Wirtschaftsrechts 2000
Manfred H. Dauses (Hg.): **Handbuch des EU-Wirtschaftsrechts.** C.H. Beck, München 2000. [zit. mit Bearbeiter und Handbuch]

Hardin 1968
Garrett Hardin: **The Tragedy of the Commons.** In: Science Vol. 162 (1968), pp 1243-1248.

Hart/Holmes/Reid 2000
Robert Hart, Peter Holmes, John Reid: **The Economic Impact of Patentability of Computer Programs.** Brüssel, October 2000, im Internet: http://europa.eu.int/comm/internal_market/en/intprop/indprop/studyintro.htm (30.10.2000).

Harhoff/Reitzig
Dietmar Harhoff, Markus Reitzig: **Strategien zur Gewinnmaximierung bei der Anmeldung von Patenten − wirtschaftliche und rechtliche Aspekte als Entscheidungsgrößen beim Schutz von F&E.** Im Internet: http://www.inno-tec.bwl.uni-muenchen.de/download/forschung/ZfB%201999%20Artikel%20_v1_.-pdf (1.8.2000).

Hellfeld 1989
Axel v. Hellfeld: **Sind Algorithmen schutzfähig?** In: GRUR 7/1989, S. 471-485.

Hoeren/Schumacher 2000
Thomas Hoeren, Dirk Schumacher: **Verwendungsbeschränkungen im Softwarevertrag. Überlegungen zum Umfang des Benutzungsrechts für Standardsoftware.** In: CR 3/2000, S. 137-146.

Holt/Morgan 1994
William H. Holt, Rockie J. Morgan: **UNIX. An Open Systems Dictionary.** Resolution Business Press, Inc., Bellewue, WA, 1994.

Horns 2000
Axel H. Horns: **Der Patentschutz für softwarebezogene Erfindungen im Verhältnis zur «Open Software»-Software.** In: JurPC v. 6. und 13.11.2000, JurPC Web-Dok. 223/2000, Abs. 1-80, im Internet: http://www.jurpc.de/aufsatz/20000223.htm (7.11.2000).

Horns 2001
Axel H. Horns: **Anmerkungen zu begrifflichen Fragen des Softwareschutzes.** Erscheint in: GRUR, Anfang 2001.

Hubmann 1987
Heinrich Hubmann: **Urheber- und Verlagsrecht.** 6. Aufl., C.H. Beck, München 1987.

Hubmann/Göttling 1998
Heinrich Hubmann, Horst-Peter Göttling: **Gewerblicher Rechtsschutz. Ein Studienbuch.** 6. Aufl., C.H. Beck, München 1998.

Hübner 1996
Claudia Hübner: **Softwareschutz: Die Debatte im internationalen Vergleich.** In: Mitteilungen der deutschen Patentanwälte 1996, S. 384ff.

Hughes 1995
Larry J. Hughes, Jr.: **Actually Useful Internet Security Techniques.** New Riders Publishing, Indianapolis, IN, 1995.

Hummer/Weiß 1997
Waldemar Hummer, Friedl Weiss: **Vom GATT ´47 zur WTO ´94. Dokumente zur alten und neuen Welthandelsordnung.** Nomos, Baden-Baden 1997.

Jander 2000
Dieter Jander: **Ist die amerikanische Regel bezüglich der Technizität besser als die deutsche?** In: Mitteilungen der deutschen Patentanwälte 2000, S. 346ff.

Kitch 1998
Edmund W. Kitch: **Patents.** In: Palgrave Dictionary of Economics & Law. Vol. 3, p. 13ff.

Klein 1998
Benjamin Klein: **Tying.** In: Palgrave Dictionary of Economics & Law. Vol. 3, p. 630ff.

Koch 2000 a
Frank A. Koch: **Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software.** In: CR 5/2000, S. 273ff.

Koch 2000 b
Frank A. Koch: **Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software.** In: CR 6/2000, S. 333ff.

Koerner 2000
Brendan I. Koerner: **The World's Most Secure Operating System.** In: The Standard, August 14, 2000, im Internet: http://www.thestandard.com/article/display/0,1151,17541,00.html (18.10.2000).

Köhntopp/Köhntopp/Pfitzmann 2000
Kristian Köhntopp, Marit Köhntopp, Andreas Pfitzmann: **Sicherheit durch Open Source? Chancen und Grenzen.** In: DuD 9/2000, S. 508-513.

Kolle 1973/1974
Gerd Kolle: **Der Rechtsschutz von Computerprogrammen aus nationaler und internationaler Sicht.** In: GRUR 1973, S. 611ff; GRUR 1974, S. 7ff.

Kolle 1977
Gerd Kolle: **Technik, Datenverarbeitung und Patentrecht. Bemerkungen zur Dispositionsprogramm-Entscheidung des Bundesgerichtshofs.** In: GRUR 1977, S. 58ff.

Kraßer 1986
Rudolf Kraßer: **Lehrbuch des Patentrechts.** 4. Aufl., C.H. Beck, München 1986.

Krempl 2000
Stefan Krempl: **Der grüne Weg in die Infogesellschaft.** In: Telepolis v. 18.10.2000, im Internet: http://www.heise.de/tp/deutsch/inhalt/te/8944/1.html (19.10.2000).

Kürten 1998
Oliver Kürten: **Spezialisierte Labors betreiben die Forschung gegen zerstörerische Programme. Computerviren spielen mal Verstecken, mal Hase und Igel mit den Scannern.** In: Computer Zeitung 4/1998, S. 16.

Jones 1968
Cacy V. Jones: **Software Is Unpatentable.** New York Times v. 23.10.1968.

Langenbach 2001
Christian Langenbach (Hg.): **Elektronische Signaturen: Kulturelle und moralische Beherrschbarkeit.** Erscheint im Springer Verlag, Berlin u.a., Frühjahr 2001.

Landes/Posner 1989
William M. Landes, Richard A. Posner: **An Economic Analysis of Copyright Law.** In: Journal of Legal Studies. Vol. XVIII (2) June 1989, p. 325ff.

Laurie/Laurie 1999
Ben Laurie, Peter Laurie: **Apache. The Definitive Guide.** 2nd ed., O'Reilly & Ass., Sebastopol, CA, 1999.

Lehmann 1983
Michael Lehmann: **Eigentum, geistiges Eigentum und gewerblichen Schutzrechte.** In: GRUR Int. 1983, S. 356ff.

Leistner/ Bettinger 1999
Matthias Leistner, Torsten Bettinger: **Creating Cyberspace. Immaterialgüterrechtlicher und wettbewerbsrechtlicher Schutz des Webdesigners.** In: Beilage zu CR 12/1999.

Lerner/Tirole 2000
Josh Lerner, Jean Tirole: **The Simple Economics of Open Source.** February 25, 2000, im Internet: http://papers.nber.org/papers/W7600 (3.12.2000).

Lesshafft/Ulmer 1988
Karl Lesshafft/Detlev Ulmer: **Softwarefehler und Gewährleistung.** In: CR 1988, S. 813-818.

Lessig 1999
Lawrence Lessig: **Code and Other Laws of Cyberspace.** : Basic Books, New York 1999.

Lessig 2000 a
Lawrence Lessig: **Europe´s 'me-too' patent law.** Financial Times online v. 11.7.2000, im Internet: http://news.ft.com/ft/gx.cgi/ftc?pagename=View&c=Article&cid=FT36QMGXJAC (3.12.2000).

Lessig 2000 b
Lawrence Lessig: **Government Property. The Bureaucrats in Washington Don't Just Break Monopolies. They Also Make Them.** In: The Standard, October 27, 2000, im Internet http://www.thestandard.com/article/display/0,1151,19762,00.html (1.11.2000).

Lessing/Weese 1995
Günter Lessing, Eckart Weese: **Organisationsstrukturen des IV-Sicherheitsprozesses.** In: Pohl/Weck 1995, S. 9-42.

Liebowitz/Margolis 1998
S. J. Liebowitz, Stephen E. Margolis: **Network Effects and Externalities.** In: Palgrave Dictionary of Economics & Law. Vol. 2, p. 671ff.

Liikanen 2000
Ari Liikanen: **Trust and Security in Electronic Communications: the European Contribution.** Vortrag auf der ISSE 2000, Barcelona, September 29 th, 2000.
http://europa.eu.int/comm/commissioners/liikanen/speeches/051099_en.htm (4.10.2000).

Lipsey/Chrystal 1999
Richard G. Lipsey, K. Alec Chrystal: **Principles of Economics.** 9th ed., Oxford University Press, Oxford 1999.

Locke 2000
Christopher Locke (ed.): **The Cluetrain Manifesto: The End of Business as Usual.** Perseus Books, Cambridge, MA, 2000.

Lohr 2000
Steve Lohr 2000: **Code Name: Mainstream.** New York Times on the Web v. 28.8.2000.

Loshin 2000
Pete Loshin: **Securing Linux. Is Open Source Too Open for Its Own Good?** In: Information Security Magazin, February 2000, im Internet: http://www.infosecuritymag.com/feb2000/Linux.htm (22.9.2000).

Lutterbeck 2000
Bernd Lutterbeck: **Globalisierung des Rechts − Am Beginn einer neuen Rechtskultur?** CR 1/2000, S. 65ff.

McKusick 1999
Marshall Kirk McKusick: **Twenty Years of Berkeley Unix: From AT&T−Owned to Freely Redistributable.** In: DiBona/Ockman/Stone 1999, pp 31-46.

McKenzie 2000
Richard B. McKenzie: **Trust on Trial. How the Microsoft Case is Reframing the Rules of Competition.** Perseus Publishing, Cambridge, MA, 2000.

Magaziner 1999
Ira Magaziner: **Balancing Public and Private Interests Through Self-Regulation.** Im Internet: http://www.stiftung.bertelsmann.de/internetcontent/deutsch/content/c2230.htm (14.3.2000)..

Manifesto 1994
Pamela Samuelson, Randall Davis, Mitchell D. Kapor, J.H. Reichman: **A Manifesto Concerning the Legal Protection of Computer Programs.** Symposium: Toward a third intellectual property paradigm. Columbia Law Review 94 (1994) no 8, p. 2308ff.

Marti 2000
Don Marti: **Focus: Security.** In: Linux Journal, 10/2000, p. 91.

Melullis 2000
Klaus J. Melullis: **Statement auf der Tagung «Wem gehört das Wissen? Geistiges Eigentum im Zeitalter des Internet»**, veranstaltet von der Heinrich-Böll-Stiftung am 20./21.10.2000 in Berlin. [Berlin 2000]

Metzger/Jäger 1999
Axel Metzger, Till Jäger: **Open Source Software und deutsches Urheberrecht.** In: GRUR 1999, S. 839ff. (30.10.2000).

Moglen 1999
Ebden Moglen: **Anarchism triumphant: Free Software and the Death of Copyright.** In: First Monday Vol. 4 (1999) Issue 8, im Internet: http://firstmonday.org/issues/issue4_8/moglen/ (30.10.2000).

Mowshowitz 1998
Abbe Mowshowitz: **Virtual Feudalism.** In: Peter J. Denning, Robert, M. Metcalfe (ed.): **Beyond Calculation. The Next Fifty Years of Computing.,** S. 213-231, Copernicus (Springer), New York: 1998.

Müller 2000
Bernhard Müller: **Künftige EG-Richtlinie über Patentierbarkeit von Computerprogrammen.** In: CRi 1/2000, S. 17f.

Münch/Kunig 1992
Ingo von Münch, Philip Kunig (Hrsg.): **Grundgesetzkommentar.** Band 1: Präambel bis Art. 20. 4., 4. Auflage, C.H. Beck, München 1992.

NCES 1998
National Center for Education Statistics: **Safeguarding Your Technology. Practical Guidelines for Electronic Education Information Security.** NCES Publication 98-297, im Internet: http://nces.ed.gov/pubs98/safetech/chapter7.html (3.12.2000).

Nüttgens/Tesei 2000 a
Markus Nüttgens, Enrico Tesei: **Open Source: Konzept, Communities und Instittutionen.** Forschungsberichte des Instituts für Wirtschaftsinformatik, Universität des Saarlandes. IWi-Heft 156, Saarbrücken 2000, im Internet: http://www.iwi.uni-sb.de/iwi-hefte/heft156.pdf (15.9.2000).

Nüttgens/Tesei 2000 b
Markus Nüttgens, Enrico Tesei: **Open Source: Produktion, Organisation und Lizenzen.** Forschungsberichte des Instituts für Wirtschaftsinformatik, Universität des Saarlandes. IWi-Heft 157, Saarbrücken 2000, im Internet: http://www.iwi.uni-sb.de/iwi-hefte/heft157.pdf (15.9.2000).

Nüttgens/Tesei 2000 c
Markus Nüttgens, Enrico Tesei: **Open Source: Marktmodelle und Netzwerke.** Forschungsberichte des Instituts für Wirtschaftsinformatik, Universität des Saarlandes. IWi-Heft 158, Saarbrücken 2000, im Internet: http://www.iwi.uni-sb.de/iwi-hefte/heft158.pdf (15.9.2000).

OFT 2000
Office of Fair Trading: **E-Commerce And Its Implications For Competition Policy.** A report compiled by the Frontier Economics Group for the Discussion Paper OFT308 Office of Fair Trading, im Internet: http://www.oft.gov.uk/html/research/reports/oft308.pdf (3.12.2000).

OSD 2000
Open Source Definition, Version 1.7,
im Internet: http://www.opensource.org/osd.html (3.12.2000).

Ostrom 1999
Elinor Ostrom: **Die Verfassung der Allmende.** Mohr, Tübingen 1999.

Palgrave Dictionary of Economics and the Law 1998
Peter Newman (ed): **The New Palgrave Dictionary of Economics and the Law.** Vol 1, 2, 3; McMillan Reference Limited, London and Basingstoke 1998, Stockton Press, New York 1998.

Paoli 2000
Nicola de Paoli: **101 Köpfe der New Economy: Wolfgang Tauchert.** In: Financial Times Deutschland v. 30.10.2000.

Patterson 1968
Lyman Ray Patterson: **Copyright in Historical Perspective.** Vanderbilt University Press, Nashville 1968.

Pipkin 2000
Donald L. Pipkin: **Information Security. Protecting the Global Enterprise.** Prentice Hall PTR, Upper Saddle River, NJ, 2000.

Pfeiffer 2000
Axel Pfeiffer: **"Davids können mit Goliath verhandeln, weil sie ein interessantes Patent halten"**. Interview in Computer Zeitung 37/14.9.2000, S. 21.

Pfaffenberger 1999
Bryan Pfaffenberger: **The Coming Software Patent Crisis: Can Linux Survive?** In: Linux Journal 10.8.1999, http://www2.linuxjournal.com/articles/currents/003.html (8.10.2000).

Pohl/Weck 1995
Hartmut Pohl, Gerhard Weck (Hrsg.): **Beiträge zur Informationssicherheit: Strategische Aspekte der Informationssicherheit und staatliche Reglementierung.** R. Oldenbourg Verlag, München, Wien 1995.

Posner 1998
Richard A. Posner: **Privacy.** In: Palgrave Dictionary of Economics & Law. Vol. 3, p. 103ff.

Potter 2000
Shawn W. Potter: **Opening Up to Open Software.** In: 6 Richmond Journal of Law & Technology, 24 (Spring 2000), im Internet http://www.richmond.edu/jolt/v615/article3.html (30.9.2000).

Quéau 1997
Philippe Quéau: **Wenn der Cyberspace zum Privatbesitz wird.** In: Le Monde Diplomatique v. Februar 1997.

Quéau 2000
Philippe Quéau: **Wem gehört das Wissen?** In: Le Monde Diplomatique v. Februar 2000.

Raden 1995
Lutz van Raden: **Die Informatische Taube. Überlegungen zur Patentfähigkeit informationsbezogener Erfindungen.** GRUR 1995, S. 451-458.

Raskind 1998
Leo J. Raskind: **Copyright.** In: Palgrave Dictionary of Economics & Law. Vol. 1, p. 478ff.

Raymond 1999
Eric S. Raymond: **The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary.** O'Reilly, Sebastopol u.a. 1999.

Rechenberg/Pomberger 1997
Peter Rechenberg, Gustav Pomberger: **Informatik-Handbuch.** Carl Hanser Verlag, München, Wien 1997.

Rehbinder 1998
Manfred Rehbinder: **Urheberrecht.** 10. Aufl., C.H. Beck, München 1998.

Reichman 1992
Jerome H. Reichman: **Legal Hybrids between the Patent and Copyright Paradigms.** In: Altes/Dommering/Hugenholtz/Kabel 1992, p. 325ff.

Reichman 1994
Jerome H. Reichman: **Legal Hybrids between the Patent and Copyright Paradigms.** In: Columbia Law Review Vol. 94 (1994) No. 8, p. 2432ff.

Reidenberg 1998
Joel Reidenberg: **Lex Informatica: The Formulation of Information Policy Rules through Technology.** In: Texas Law Review, Vol. 76 No. 3 (February 1998), p. 553ff.

Richter/Furubotn 1999
Rudolf Richter, Eirik G. Furubotn: **Neue Institutionenökonomik.** 2. Aufl., Mohr, Tübingen 1999.

Rittner 1999
Fritz Rittner: **Wettbewerbs- und Kartellrecht.** 6. Aufl. C.F. Müller, Heidelberg 1999.

Rivette/Kline 2000
Kevin G. Rivette, David Kline: **Wie sich aus Patenten mehr herausholen lässt.** In: Harvard Business manager 4/2000, S. 29ff.

Rivette/Kline 1999
Kevin G. Rivette, David Kline: **Surviving a War with Patents.** In: UpsideToday v. 10.12.1999, http://www.upside.com/texis/mvm/opinion/story?id=382a24f90 (8.10.2000).

Rötzer 1999
Florian Rötzer: **Sollen Gesetzestexte kostenlos vom Staat ins Internet gestellt werden?** Bericht vom EDV-Gerichtstag 1999, im Internet: http://www.heise.de/tp/deutsch/inhalt/te/5317/1.html (2.12.2000).

Rosenoer 1997
Jonathan Rosenoer: **Cyberlaw. The Law of the Internet.** Springer-Verlag, New York 1997.

Rothe 1993
Lother Rothe: **Produkthaftung. Rechtliche Grundlagen und ihre Auswirkungen auf die Industrie.** In: CR 5/1993, S. 310-313.

Ruppelt 1988
Martin Ruppelt: **Vertragseinheit und Softwaremängel. Anmerkung zum BGH Urteil vom 04.11.1987 - VIII ZR 314/86.** In: CR 1988, S. 994-995.

Sabbatini 2000
Pierluigi Sabbatini: **The Microsoft Case.** In: Social Science Research Network Electronic Paper Collection, im Internet: http://papers.ssrn.com/paper.taf?abstract_id= 223769 (15.8.2000).

Samuelson 2000
Pamela Samuelson: **Economic and Constitutional Influences on Copyright Law in the United States.** In: United States Intellectual Property Law. Hugh Hansen (ed.), Sweet & Maxwell, forthcoming 2000, im Internet: http://www.sims.berkeley.edu/~pam/papers/Sweet&Maxwell_1.htm (15.9.2000).

Shankerman 1998
Mark Shankerman: **How Valuable is Patent Protection? Estimates by Technology Field.** RAND Journal of Economics Vol. 29, No. 1. Spring 19998, pp 77-107.

Schickedanz 2000
Willi Schickedanz: **Das Patentierungsverbot von «mathematischen Methoden», «Regeln und Verfahren für gedanklichen Tätigkeiten» und die Verwendung mathematischer Formeln im Patentanspruch.** In: Mitteilungen der deutschen Patentanwälte 2000, S. 173ff.

Schily 1999
Otto Schily: **Weltweite Kommunikation − eine neue Kultur gemeinsamer Verantwortung**, im Internet: http://www.stiftung.bertelsmann.de/internetcontent/deutsch/content/c2230.htm (14.3.2000).

Schmid 2000
Herbert Schmid: **Open Source Hardware. Hardware-Download aus dem Internet.** In: c't 22/2000, S. 294-300.

Schmidt 1999
Ingo Schmidt: **Wettbewerbspolitik und Kartellrecht.** 6. Aufl., Lucius & Lucius, Stuttgart 1999.

Schneier 1999
Bruce Schneier: **Crypto-Gram, September 15,1999.** Im Internet: http://www.counterpane.com/crypto-gram-9909.html (8.10.2000).

Schneier 2000
Bruce Schneier: **Secrets & Lies. Digital Security in a Networked World.** John Wiley & Sons, Inc., New York u.a. 2000.

Schricker 1999
Gerhard Schricker: **Urheberrecht. Kommentar.** 2. Aufl., C.H. Beck, München 1999. [zit. mit §§ und den jeweiligen Bearbeitern]

Schwartz 1996
Hillel Schwartz: **The Culture of the Copy.** Zone Books, New York 1996.

Schwarz 2000
Gerhard Schwarz: **Kontroversen über Drogen und Patente. Bericht von der Jahrestagung der Mont Pèlerin Society.** In: Neue Zürcher Zeitung v. 25.11.2000.

Schwebel 2000
Robert Schwebel: **The Good, The Bad and The Ugly. Gefährliche Treiber-Entwicklungstools.** Linux-Magazin 11/2000, S. 140-142.

Schyguda 1995
Georg Schyguda: **Bewertung der Sicherheitskonzepte der digitalen Mobilkommunikation.** In: Pohl/Weck, S. 65-98.

Searls 2000
Doc Searls: **World Domination? Heh.** In: Linux Journal online, im Internet: http://www2.linuxjournal.com/articles/tradeshow/0024.html (16.11.2000).

Seeger 2000
Jürgen Seeger: **Trends zum 3. Breiter Einsatz freier Software.** In: iX 11/2000, S. 10-12.

Seifried 2000
Kurt Seifried: **PAM - Pluggable Authentication Modules.** In: SysAdmin Magazine 9/2000, p. 8ff.

Senti 2000
Richard Senti: **WTO. System und Funktionsweise der Welthandelsordnung.** Schulthess, Zürich 2000.

Shapiro/Varian 1999
Carl Shapiro, Hal R. Varian: **Information Rules. A Strategic Guide to the Network Economy.** Harvard Business School Press, Boston, MA, 1999.

Siepmann 1999
Jürgen Siepmann: **Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software.** In: JurPC, JurPC Web-Dok. 163/1999, Abs 1-289, im Internet: http:///jura.uni-sb.de/jurpc/aufsatz/19990163.htm (18.9.1999).

Slind-Flor 1999
Victoria Slind-Flor: **Gold Diggers.** In: LawNewsNetwork.com v. 20.12.1999, im Internet: http://www.lawnewsnetwork.com/practice/iplaw/news/A11873-1999Dec17.html (8.10.2000).

Smarr/Graham 2000
Larry Smarr, Susan Graham: **Recommendations of the Panel on Open Source Software for High End Computing.** President's Information Technology Advisory Committee (PITAC), September 11, 2000, im Internet: http://www.ccic.gov/ac/pres-oss-11sep00.pdf (18.10.2000).

Stenz 2000
Thomas Stenz: **Vormarsch des Immateriellen. Anpassung der Finanz-Berichterstattung in der New Economy.** In: Neue Zürcher Zeitung vom 27.7.2000.

Stahlknecht 1993
Peter Stahlknecht: **Einführung in die Wirtschaftsinformatik.** 6. Aufl., Springer-Verlag, Berlin, Heidelberg 1993.

Stallman 1999
Richard Stallman: **The GNU Operation System and the Free Software Movement.** In: DiBona/Ockman/Stone 1999, pp 53-70.

Stallman 2000
Richard Stallman im **E-Mail-Briefwechsel mit Jorrit Tyberghein**, im Internet: http://crystal.linuxgames.com/rms.html (24.10.2000).

Straus 1998
Joseph Straus: **Patente als Stütze der modernen Biotechnologie.** In: Spektrum der Wissenschaft 4/1998, S. 28-36.

Taeger 1996
Jürgen Taeger: **Produkt- und Produzentenhaftung bei Schäden durch fehlerhafte Computerprogramme.** In: CR 5/1996, S. 256-270.

Tauchert 1999
Wolfgang Tauchert: **Patentschutz für Computerprogramme.** In: GRUR 1999, S. 829ff.

Tauchert 2000
Wolfgang Tauchert: **Grundlagenpapier für die Tagung «Wem gehört das Wissen? Geistiges Eigentum im Zeitalter des Internet»**, veranstaltet von der Heinrich-Böll-Stiftung am 20./21.10.2000 in Berlin. [Berlin 2000]

Ticehurst 2000
Jo Ticehurst: **Windows 2000 'Not Ready for Web Servers'. Linux Becoming Most Secure OS - Gartner.** In: Linux Today v. 7. November 2000, im Internet: http://linuxtoday.com/news_story.php3?ltsn=2000-11-07-001-06-SC-MR-MS (7.11.2000).

Torvalds 1999
Linus Torvalds: **The Linux Edge.** In: Dibona/Ockman/Stone 1999, S. 101-112.

Turner 2000
Colin Turner: **The Information e-Conomy. Business Strategies for Competing in the Digital Age.** Kogan Page, Ltd., London 2000.

Ullrich/Konrad 2000
Hanns Ulrich, Karlheinz Konrad: **Gewerblicher Rechtsschutz.** In: Handbuch des EU-Wirtschaftsrechts 2000.

Valentine 2000
Debra A. Valentine: **Abuse of Dominance in Relation to Intellectual Property: U.S. Perspectives and the INTEL Case.** In: CRi 3/2000, S. 73ff.

Varian 1999
Hal Varian: **Grundzüge der Mikroökonomik.** 4. Aufl., R.Oldenbourg, München, Wien 1999.

Vixie 1999
Paul Vixie: **Software Engineering.** In: DiBona/Ockman/Stone 1999, pp 91-100.

Voßbein 1995
Reinhard Voßbein: **Eigenverantwortlichkeit und Marktwirtschaft als Steuerungsimpulse der IT-Sicherheit. Sicherheit - ein Trivialproblem?** In: Pohl/Weck 1995, S. 43-50.

Wayner 2000
Peter Wayner: **Whose Intellectual Property Is It, Anyway? The Open Source War.** New York Times on the Web v. 24.8.2000.

Wenger/Snyder 2000
Etienne C. Wenger, William M. Snyder: **Communities of Practice: Warum sie eine wachsende Rolle spielen.** In: Harvard Business manager 4/2000, S. 55-62.

Winischhofer 2000
Thomas Winischhofer: **Computersoftware und Patentrecht.** Dissertation. Im Internet über: http://Webit.com/tw/patent.shtml (1.9.2000).

Young/Goldman Rohm 2000
Robert Young, Wendy Goldman Rohm: **Der Red Hat Coup.** MITP, Bonn 2000.

Ohne Verfasser

Bereitschaft zur Anzeige bei Computerdelikten nimmt ab.
In: Computer Zeitung 31 v. 6.8.1998.

Heise Newsticker vom 15.9.2000
US-Regierung soll Open Source fördern. In: Heise Newsticker v. 15.9.2000, im Internet: http://www.heise.de/newsticker/data/odi-15.09.00-001/ (8.10.2000).

Zeitschriften

CR
Computer und Recht, Verlag Otto Schmidt, Köln

CRi
Computer und Recht International, Verlag Otto Schmidt, Köln

c't
c't Magazin für Computertechnik, Verlag Heinz Heise, Hannover

GRUR
Gewerblicher Rechtschutz und Urheberrechtschutz, Verlag C.H.Beck, München

GRUR int
Gewerblicher Rechtschutz und Urheberrechtschutz, Internationaler Teil, Verlag C.H.Beck, München

iX
iX Magazin für Professionelle Informationstechnik, Verlag Heinz Heise, Hannover

Rechtsprechungsnachweise

BGH «Dispositionsprogramm» (1977)
GRUR 1977, S. 96ff.

BGH «Logikverifikation» (2000)
BGH Beschluss vom 13.12.1999 − X ZB 11/98,
JurPC Web-Dok 72/2000, Abs.1-37,
im Internet: http://www.jurpc.de/rechtspr/20000072.htm (2.12.2000);
Mitteilungen der deutschen Patentanwälte **91** (2000), Nr. 6, S. 293-297.

BGH «Rote Taube» (1969)
GRUR 1969 Nr. 12, S. 672-676.

BGH «Sprachanalyseeinrichtung» (2000)
Mitteilungen der deutschen Patentanwälte **91** (2000), Nr. 8, S. 359-361;
GRUR Int. 2000 Nr. 10, S. 930-936.

BGH «Tauchcomputer» (1992)
GRUR 1992, S. 430ff.

BPatG «Programm für Datenverarbeitungsanlage als solches» (2000)
Bundespatentgericht Beschluss vom 28. Juli 2000 − 17 W (pat) 69/98,
JurPC Web-Dok 195/2000, Abs.1-50,
im Internet: http://www.jurpc.de/rechtspr/20000195.htm (2.12.2000).

Technische Beschwerdekammer d. EPA "Computerprogrammprodukt / IBM" (1998)
Entscheidung der Technischen Beschwerdekammer 3.5.1 vom 1. Juli 1998 − T 1173/97-3.5.1,
im Internet: http://www.european-patent-office.org/epo/pubs/oj99/10_99/10_6099.pdf (2.12.2000).

Technische Beschwerdekammer d. EPA "Controlling pension benefits system / PBS PARTNERSHIP" (2000)
Entscheidung der Technischen Beschwerdekammer 3.5.1 vom 8. September 2000 −
T 0931/95–3.5.1, im Internet: http://www.european-patent-office.org/dg3/pdf/t950931eu1.pdf
(2.12.2000).

State Street (1998)
US Court of Appeals for the Federal Circuit: Patentability of Date Processing Systems,
State Street Bank & Trust Co. v. Signature Financial Group, Inc. July 23, 1998,
CRi 1/2000, S. 19ff.
Im Internet: http://www.law.emory.edu/fedcircuit/july98/96-1327.wpd.html (2.12.2000).

Junger v. Daley (2000)
United States Court of Appeals for the Sixth Circuit in re Peter D. Junger, Plaintiff-Appellant, v. William Daley, United States Secretary of Commerce, et al., Defendants-Appellees. No. 98-4045, April 4, 2000.
Im Internet: http://pacer.ca6.uscourts.gov/cgi-bin/getopn.pl?OPINION=00a0117p.06 (3.9.2000).