Software−Patente im Spiegel von Softwareentwicklung und Open Source Software

von Robert A. Gehring und Bernd Lutterbeck

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 in the books’ und dem ‘law in action’. Unbeschadet des Wortlauts des deutschen und amerikanischen Patentrechts haben sich Gerichte und Patentämter in Europa und den USA nicht gescheut, ‘Software−Patente’ rechtlich anzuerkennen. Das Ob von "Software−Patenten" ist nicht mehr ernsthaft bestritten. Die Frage ‘Ist Software patentierbar?’ ist praktisch längst beantwortet − "a matter for the history books"[1].

Darüberhinaus bestand in der europäischen Literatur weitgehende Einigkeit insofern, als daß die Patentierungspraxis in Europa restriktiver ausfallen müsse als die des vielkritisierten amerikanischen Patentamtes. Insbesondere bestand so etwas wie ein Konsens, daß reine Geschäftsmethoden nicht patentierbar sein sollten.[2] Auch diese Sicht der Literatur scheint gefallen zu sein. Das Europäische Patentamt in München hat kürzlich der Firma Amazon ein Patent erteilt, das praktisch alle Computer−basierten Verfahren der bestellten Lieferung von Geschenken an Dritte umfaßt (und mehr). Dieses Patent ist ein Abkömmling des in den USA erteilten "1−Click" Patentes, weist aber einen breiteren Anspruchsbereich als das Original auf. [3]

---

[1] Cohen & Lemley, 2001, S. 1.

[2] In diesem Sinne hatte sich 2001 auch die EU−Kommission in ihrem Konsultationspapier und ihrem Vorschlag für eine Richtlinie zu Computer−implementierten Erfindungen (COM(2002)92) geäußert. Auch die zuständige Berichterstatterin des Rechtsausschusses des EU−Parlamentes, Arlene McCarthy, hat diesen Aspekt immer wieder betont, wenn auch in dem von ihr vorgelegten Richtlinienentwurf nicht konsequent umgesetzt, wie ihr jedenfalls Kritiker vorhalten. Vgl. etwa die Äußerungen des FFII unter http://swpat.ffii.org/papers/eubsa-swpat0202/juri0304/.

[3] Pat. Nr. EP0927945, 1998 ursprünglich angemeldet als "Method and system for placing a purchase order via a communications network"; vgl. dazu die Presserklärung des FFII München vom 15.8.2003 mit weiterführenden Links,

Gleichwohl, oder vielleicht auch gerade deswegen, hat das Thema ‘Software−Patente’ gegenwärtig Konjunktur. Trotz aller Schwierigkeiten der Abgrenzung gehen alle Beteiligten diesseits und jenseits des Atlantiks von der großen praktischen und ökonomischen Relevanz des Themas aus.[4] Dabei bilden alte Argumente und einige 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.
  Möglicherweise neu sind die Antworten, die eine wachsende Zahl sog. freier Softwarentwickler auf die alten Fragen gibt.

- 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 etablierten Fachöffentlichkeit und getragen von Akteuren, die fast nie Ökonomen oder Juristen, sondern häufig Technologen oder Informatiker sind.[5]

- Neu und in Deutschland noch nirgends diskutiert sind Zahlen über die weltweite Verteilung von Free/Open Source−Entwicklern. Deutsche Entwickler bilden danach die zweitgrößte Gruppe. Insgesamt sind europäische Entwickler dominierend. Diese Befunde sind so bemerkenswert,

---

http://swpat.ffii.org/neues/03/amaz0818/index.de.htm.

[4] Die Relevanz des Themas wird aktuell durch die Entwicklungen im Fall Eolas v. Microsoft deutlich gemacht, in dem es um den Gültigkeitsbereich des U.S. Patents Nr. 5,838,906 geht. Vgl. die Mitteilung des W3−Konsortiums unter http://www.w3.org/2003/08/patent [01 Sep 2003]. Das genannte Patent ("Distributed hypermedia method for automatically invoking external application providing interaction and display of embedded objects within a hypermedia document") wurde am 17. November 1998 erteilt und könnte erhebliche Auswirkungen auf die Benutzbarkeit des Internets haben. Microsoft hat bereits Überlegungen bekannt werden lassen, denen zufolge die Architektur des Internet Explorers (immerhin mit ca. 85% Marktanteil der meistgenutzte Webbrowser) erheblich modifiziert werden könnte.

[5] Deutsche Juristen übersehen häufig, dass die meistgenutzte Lizenz für quelloffene Software, die GNU Public License (GPL), von dem US−Informatiker Richard Stallman entwickelt wurde.

dass eine Erklärung mit den bekannten ökonomischen Modellen zunächst versagt.[6]

- Ganz sicher neu sind Argumente, die Fachleute aus dem Bereich der Sicherheit der Informationstechnologie (IT-Sicherheit) für die Offenlegung von Software ins Feld führen.[7]

Vor allem die letztgenannten Gesichtspunkte geben der aktuellen Debatte um Computer−implementierte Erfindungen ein Gewicht, das den Kern der neuen Ökonomie betrifft wird: Eine Internet−gestützte Ökonomie steht und fällt mit der Lösung von Problemen der Sicherheit und Verläßlichkeit der Netzwerke und der darin eingesetzten Software. Als (angewandte) Informatiker sind wir besorgt darüber, daß die überwiegend juristisch−dogmatisch geführte Debatte an den fachlichen Problemen der Informatik vorbeigeht und so ein Innovationen nicht förderliches Umfeld erzeugt. Wir haben deshalb in den letzten Jahren in einer Reihe von Beiträgen versucht, informatische und juristische Zugänge zueinander anschlußfähig zu machen.[8]

Im folgenden wollen wir einen praktisch wichtigen Teilaspekt[9] näher beleuchten: das Problem nämlich, bei umfangreicher Patentierbarkeit von Software unabsichtlich bestehende Patentrechte zu verletzen. Es wird aus der Sicht der Softwareentwicklung diskutiert, wie es geschehen kann, daß Patentrechte unabsichtlich verletzt werden, welche Konsequenzen das für den Entwickler, bzw. die Entwicklung von Software im allgemeinen, haben kann, und wie man den

---

[6] Diese Lücke füllt weitgehend die Monographie von Weber, 2003. Unsere eigenen empirischen Erhebungen bestätigen die Aussage, sind aber im Detail noch nicht ausgewertet. Verfügbar sind sie über http://ig.cs.tu-berlin.de/forschung/OpenSource/index.html.

[7] Die Debatte darüber hat sich in den vergangenen Monaten intensiviert. Vgl. u.a. Gehring, 2003, der die Zusammenhänge zwischen Software-Ökonomie und IT-Sicherheit diskutiert sowie Challet & Le Du, 2003, die ein theoretisches Modell der Fehlerbeseitigung bei ‘closed source’ und ‘open source’ Softwareentwicklung vorgestellt haben, das erklären könnte, wieso quelloffene Software schneller von Fehlern bereinigt wird.

[8] Vgl. u.a. Gehring, 2000, und Lutterbeck, Horns & Gehring, 2000.

[9] Hier nicht behandelt, aber von ähnlicher Bedeutung ist das Problem des ‘Reverse Engineering’ und seine weitgehende Einschränkung im Urheberrecht. Siehe dazu juristisch/ökonomisch u.a. Samuelson & Scotchmer, 2000, sowie technisch/ökonomisch van Zuylen (Hrsg.), 1993.

bestehenden Risiken in der Praxis begegnen kann. Ein besonderes Augenmerk richten wir dabei auf die Konsequenzen für die Entwickler von Freier und Open Source Software.

Softwareentwicklung als Konfigurierung einer Universalmaschine

Ausgangspunkt unserer Betrachtungen ist die Entwicklung von Standardsoftware für Standard−PCs, die sich als die Bewältigung folgender Aufgabe beschreiben läßt:

> "Configure machine M to produce effects R in domain D."[10]

Software ist dazu da, Probleme zu lösen. Diese Probleme bestehen in einer mehr oder minder genau bestimmten Problemdomäne D. Eine entwickelte Software bewirkt in einer Universalmaschine M, wie es der Standard−PC ist, eine wohldefinierte Folge von Systemzuständen, die als Programmablauf bezeichnet wird. Dabei auftretende Effekte R, d.h. konkrete Verhaltensweisen der Maschine M in Interaktion mit ihrer Umwelt, sollten deterministisch zur Lösung der ursprünglich gestellten Aufgabe führen. Andernfalls ist die Software fehlerhaft.

Vor der fertigen Lösung muß jedoch erst einmal eine Lösungsidee her, die dann in Form eines Programmes umgesetzt werden kann. Begonnen wird mit der Erhebung von Anforderungen (Requirements) an die Software, die sich sowohl aus der Problemstellung als auch der Problemdomäne mit ihren Charakteristika ergibt. Diese fließen als notwendige Bedingungen in die Lösungsidee ein. In der Lösungsidee, d.h. der mehr oder minder formalen, funktionalen Spezifikation, werden die notwendigen Effekte noch unabhängig von einer konkreten Formulierung entworfen. Anschließend wird der Lösungsweg in einer Programmiersprache so formuliert, daß der Compiler daraus die dem Computer verständlichen Anweisungen zum Programmablauf generieren kann. Abbildung 1 zeigt in einem Modell, wie diese Schritte (von Innen nach Außen) zusammenhängen.

---

[10] Kovitz, 1999, S. 25.

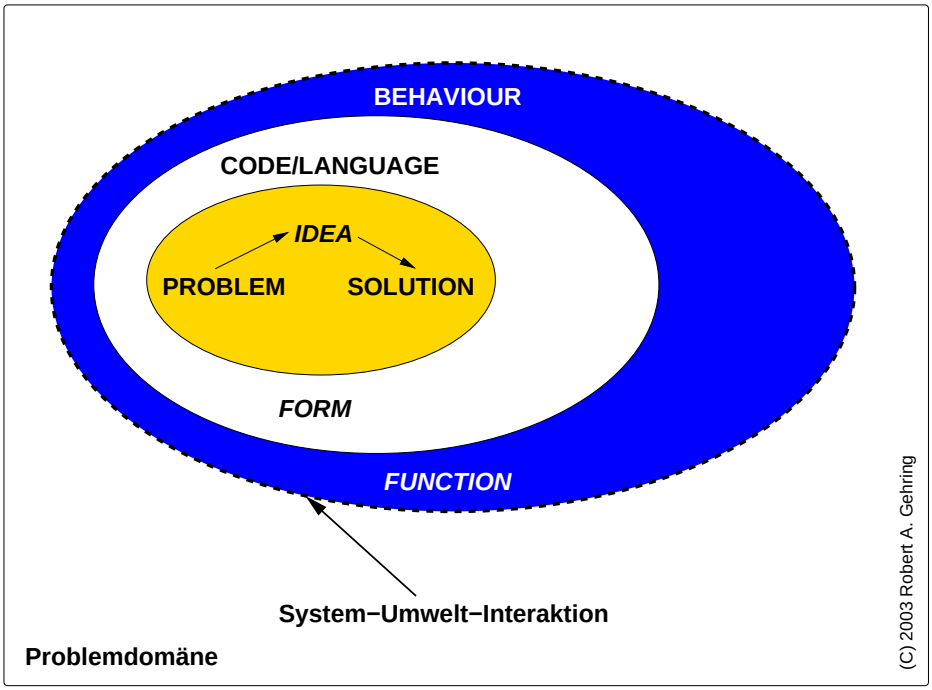

Abbildung 1: Schritte der Softwareentwicklung

So weit, so gut. Jede Informatikerin und jeder Informatiker wird mit einer solchen Vorstellung von Softwareentwicklung vertraut sein. Was es aus Sicht der hier behandelten Thematik festzuhalten gilt, ist der folgende Dreischritt: Problem → Lösung(sidee) → Lösungsformulierung. Von rein fachlicher Warte betrachtet, haben Software−Patente hier keinen Platz − ein ingenieurtechnisches Paradigma mit rechtlichen Problemen, wie wie wir sehen werden.

Patentschutz für Software

Für Software stehen prinzipiell zwei Schutzinstrumente des Immaterialgüterrechts zur Verfügung. Im Urheberrecht wird die Form von Software geschützt, d.h. der je konkrete Ausdruck der Lösungsformulierung.[11] Das Patentrecht schützt hingegen die funktionalen Aspekte z.B. einer Maschine, nicht jedoch die zugrundeliegenden Ideen als solche. Insofern Patentschutz für ein bestimmtes Verhalten eines Computers gewährt wird, ist üblicherweise die Rede von

---

[11] Vgl. UrhG §69a (1), wo allerdings selbst dem Entwurfsmaterial Urheberrechtsschutz zuerkannt wird, was aus informatischer Perspektive insofern problematisch ist, als Entwürfe immens unterschiedliche Detaillierungsgrade aufweisen können, wobei die Schwelle dessen, was überhaupt noch als individuelles Werk ansehbar ist, durchaus nicht zwingend überschritten werden muß. Allerdings war dieser sogenannte 'Schutz der kleinen Münze' von der EU ausdrücklich beabsichtigt (RL 91/250/EWG), wodurch für Software ein signifikant stärkerer Urheberrechtsschutz gewährt wird als für andere Sprachwerke.

Software−Patenten. Jedes Programm, das einen Standard−PC zu einem patentgeschützten Verhalten veranlaßt, wird dann vom Patent abgedeckt. Dabei kommt es nicht auf eine bestimmte Formulierung der Anweisungen zum Bewirken des Verhaltens an. Jede Formulierung, die ein äquivalentes Verhalten bewirkt, kann betroffen sein.[12] In der Konsequenz sind fachlich naheliegende Lösungsideen dann nicht verwendbar, wenn ihre Umsetzung in PC−Software, und deren Ablaufenlassen auf dem PC, ein Verhalten bewirkt, das patentgeschützt ist, wie in den USA der Supreme Court bereits in einer sehr frühen Entscheidung (Gottschalk vs. Benson[13]) festgestellt hat :

> "It is conceded that one may not patent an idea. But in practical effect that would be the result if the formula [...] were patented in this case. The mathematical formula invoked here has no substantial practical application execept in connection with a digital computer [...]"

Diese Einsicht eines in sich sehr widersprüchlichen Urteils blieb ohne nennenswerte Folgen. In den letzten drei Jahrzehnten wurde, ausgehend von Gerichtsentscheiden im Vereinigten Königreich und den USA,[14] in zunehmendem Maße Gegenstände für patentierbar erachtet, die alltägliche Probleme der Softwareentwicklung betreffen: Zahlenkonversion, Sortierverfahren, Datenkomprimierung, Speicherverwaltung u.v.m. Die Patentämter zogen nach, manchmal eilten sie auch voraus, und änderten ihre Richtlinien für die Patenterteilung entsprechend, was zu einem exponentiell wachsenden Fluß von Anmeldungen für ‘Software−Patente’ führte.[15]

Damit hielten Personalausstattung, Qualifikation und Recherchemöglichkeiten in den Patentämtern nicht mit, worunter die Qualtität der

---

[12] Das gilt zumindest in Ländern wie Deutschland oder USA, wo eine Äquivalenzdoktrin Geltung hat.

[13] 409 U.S. 63 (1972). In dem Fall ging es um die Konversion von Zahlen aus Binary Coded Decimal (BCD)−Darstellung in reine Binärform.

[14] Kretschmer, 2003.

[15] Eine ähnliche Entwicklung läßt sich seit Jahren im Bereich der Biopatente beobachten. Daraus ergeben sich ähnliche Probleme wie im Bereich der Software−Patente. So ist es möglicherweise kein Zufall, daß ein weltbekannter Biologe, Mark Fishman, Forschungsdirektor bei Novartis in Boston, vergleichbare Forderungen erhebt wie viele Softwareentwickler: "Als Wissenschaftler kann ich sagen, daß die gegenwärtige Einstellung zum geistigen Eigentum den Fortschritt der Wissenschaft behindert. Geistiges Eigentum ist normalerweise ein Zaun. Wir müssen ihn überspringen oder durchlässig machen." Vgl. Heuer, 2003, S. 27.

Patenterteilungsprozesse litt. Es wurden immer mehr Patente erteilt, die wohl einer strengen Prüfung nicht hätten standhalten dürfen. Dazu trägt insbesondere in den USA bei, daß das dortige Patentamt sich aus den Einnahmen finanziert, die von Patentinhabern gezahlt werden. Für Patentanmelder wurde es im Gegenzug einfacher, breitere Patentansprüche mit geringerem Aufwand zu produzieren, was zu mehr Patentanmeldungen führte usw. usf. Der ‘positive feedback’ hat den Regelkreis aus Patentanmeldern, Patentämtern und Patentgerichten fest im Griff und generiert immer neue Patente mit notwendig sinkender Qualität, ein "Patentdickicht"[16], in dem man sich schnell verirren kann.

Softwareentwickler stehen also grundsätzlich immer vor dem Problem, daß eine von ihnen entworfene Lösung Patente verletzen könnte. Es stellt sich die Frage, wie groß das Problem in der Praxis ist.[17]

Software−Patente als Problem der Softwareentwickung

Da es sich beim Patentschutz um ein gesetztlich geschütztes Exklusivrecht handelt, kann der Patenthalter jedem anderen die wirtschaftliche Verwertung des geschützen Gegenstandes, letztlich also der geschützten Idee, verbieten. Bereits in die Programmentwicklung geflossene Investitionen wären damit verloren, gibt es doch keine praktische Möglichkeit, einen Patentinhaber zur Lizenzierung seiner geschützten Ideen zu zwingen. Selbst eine in Aussicht gestellte Lizenzierung kann ökonomisch sinnlos sein, wenn die Lizenzgebühren den zu erwartenden Gewinn aus der Nutzung oder Vermarktung der Software übersteigen. Jedenfalls steigt der Druck auf denjenigen, der auf die Lizenz angewiesen ist, auch zu Bedingungen zu lizenzieren, die normalerweise inakzeptabel wären. Ohne die Lizenz kann das Produkt nicht vermarktet werden − ein ’hold−up’−Problem entsteht (Shapiro 2000).

---

[16] Shapiro, 2000.

[17] Eine Vergrößerung der Rechtsunsicherheit resultiert zudem daraus, daß die tatsächliche Wirkung des Patentschutzes, seine ‘Breite’ letztendlich nur vor Gericht, ggf. dem jeweils höchsten Gericht, festgestellt werden kann. Solche Klagen sind langwierig und teuer, so daß insbesondere kleine und mittlere Unternehmen diesen Weg scheuen werden. Die Transaktionskosten zur Erlangung von Rechtssicherheit wirken insofern zugunsten des Patentinhabers, unabhängig davon, ob das Patent überhaupt zurecht erteilt wurde.

Um eine solche ’hold−up’−Situation nach Fertigstellung der Software zu vermeiden, muß im Prinzip der gesamte Entwicklungsprozeß einer Vermeidungsstrategie unterworfen werden.[18] Das unvermutete Auftauchen eines bestehenden Patentschutzes der zu einer ’hold-up’−Situation führen könnte, wäre dann im Grunde genommen als ein fachlicher Fehler bei der Bestimmung der Requirements zu interpretieren.

Wie aus der informatischen Literatur bekannt ist, sind Fehler in dieser Phase, d.h. während der Problemanalyse, besonders kostspielig zu beseitigen und sollten deshalb möglichst vermieden werden.

Ausgehend von der entworfenen Funktionalität der Lösung muß ermittelt werden, ob Patentschutz für die Gesamtlösung oder für Teillösungen besteht. Tangiert die entworfene Lösung ein bestehendes Patent nur am Rande, so läßt sich in der Regel die Software ‘am Patentschutz vorbei’ designen. Betrifft ein Patent jedoch Basistechnologie, wie etwa elementare Kommunikations−, Verschlüsselungs−, Speicherverwaltungstechnologie usw., ist ein derartiges Ausweichen nicht mehr möglich, ohne die Funktionalität der Lösung maßgeblich zu modifizieren. Ob sie anschließend noch geeignet ist, das ursprüngliche Problem zu lösen, kann nur im Einzelfall entschieden werden.

Gelangt man zu einer solchen Erkenntnis, müssen ggf. Lizenzverhandlungen aufgenommen werden, oder die Softwareentwicklung muß abgebrochen werden, bevor unnötige Kosten entstanden sind.

---

[18] Alternativ kommt eine Verteidigungsstrategie durch Aufbau eines eigenen Patentportfolios in Frage. Eine Verteidigungsstrategie ist jedoch inhärent durch einen Zielkonflikt geprägt: Die weitaus meisten Unternehmen sehen einen strategischen Vorteil darin, schnell mit einem Produkt auf dem Markt zu sein, d.h. die ‘lead time’ ist entscheidend. Der Prozeß der Patentanmeldung und -erteilung kostet viel Zeit, wobei unklar ist, ob das in Gestalt eines Patentes gewonnene Monopol auch faktisch in einen Wettbewerbsvorteil umsetzbar sein wird. Vgl. u.a. die gemeinsame Studie des Fraunhofer Institutes für Systemtechnik und Informationstechnik, Karlsruhe, und des MPI München zu "Mikro− und Makroökonomischen Implikationen der Patentierbarkeit von Softwareinnovationen", in der festgestellt wird, daß Softwarepatente als Wettbewerbsvorteile weitaus geringere Bedeutung besitzen als der zeitliche Vorsprung, mit dem ein Produkt auf den Markt komm (‘lead time’). OECD−Studien bestätigt diese Erkenntnisse. Vgl. OECD, 1996.

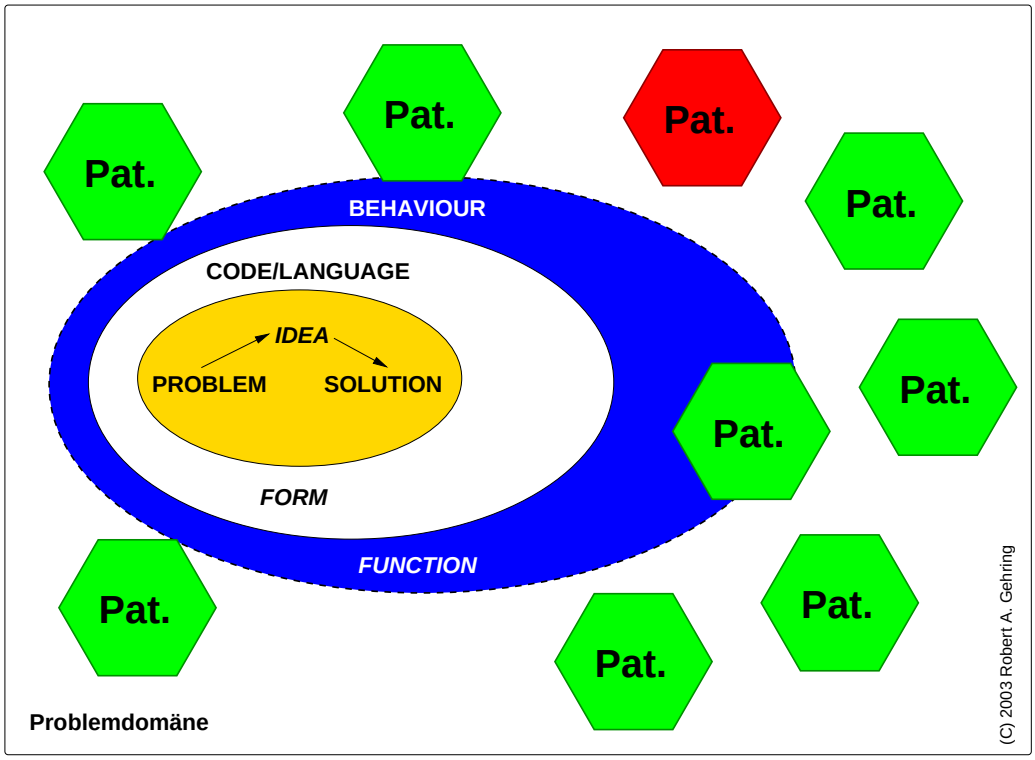

Abbildung 2: Das Patent−Problem

In Abbildung 2 sind zwei Typen von Patenten zu sehen. Die heller gezeichneten stellen jene Patente dar, die einerseits in die Problemdomäne fallen und andererseits bereits erteilt, oder angemeldet und recherchierbar sind. Der zweite Typ von Patenten, in der Grafik dunkler dargestellt, wird im folgenden Abschnitt diskutiert.

Grenzen der Patentrecherche

Der zweite Typ von Patenten, in der Abbildung dunkler gezeichnet, ist noch nicht recherchierbar, da es im Laufe des Patentanmeldungs− und −erteilungsprozesses eine Phase gibt, in der Patentanmeldungen geheimgehalten werden. Aufgrund der Spezifika der Arbeitsweise der Patentämter kann es sich um Monate bis Jahre handeln. Sollte ein Patent später erteilt werden, tritt erneut das oben bereits geschilderte Problem mit den Exklusivrechten des Patentinhabers auf: Bereits in die Softwareentwicklung geflossene Aufwendungen werden unter Umständen wertlos. Im Falle der nicht recherchierbaren Patente

ist das aber ein unvermeidliches Risiko.[19]

Wie hoch dieses Risiko ausfällt, hängt im Wesentlichen von der Granularität der erteilten Patente ab.

Granularität ist kein Terminus aus der Sprache der Patentjuristen. Mit Granularität soll hier die Größe des funktionalen Anteils an einem Softwaresystem gemeint sein, der durch ein Software−Patent abgedeckt wird. Je feiner die Granularität, desto kleiner der funktionale Anteil, und umgekehrt. Im Extremfall, bei sehr feiner Granularität, können ein paar Zeilen Quellcode (LOC) in einem zig−Millionen−LOC−Projekt von der Wirkung eines Patents erfaßt sein.[20]

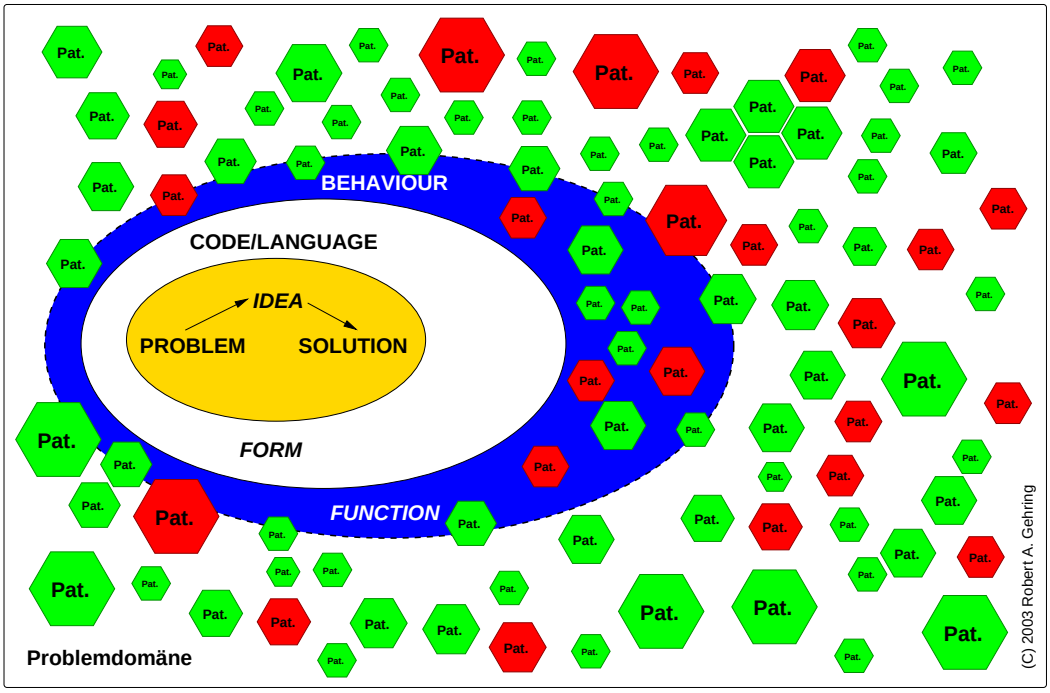

Abbildung 3: Die Patent−Gefahr

Hat man nur ein einzelnes Patent im Blick, mag das unproblematisch, ja vorteilhaft erscheinen. Hat man es mit sehr vielen Patenten feiner Granularität zu tun, stellt sich die Situation dar, wie in Abbildung 3 gezeigt: Würde man auf die patentgeschützte Funktionalität verzichten (müssen), würde ein System ähnlich

---

[19] Kritiker sprechen in solchen Fällen von "U−Boot−Patenten", da sie unverhofft auftauchen und genutzt werden (können), einen potentiellen Wettbewerber zu torpedieren.

[20] Wir adaptieren einen Begriff der Granularität von Hohmann, 2003, S. 19, der mit Granularität "the level of work performed by a component" bezeichnet.

einem ‘schweizer Käse’, mit vielen ‘funktionalen Löchern’, entstehen. Daß ein derartiges System den ursprünglich ermittelten, funktionalen Requirements gerecht werden würde, darf getrost bezweifelt werden.

Hinzu kommt das oben geschilderte Problem der (noch) nicht recherchierbaren Patentanmeldungen, die sich zu einem in der Praxis unkalkulierbarem Risiko akkumulieren können. Jede Erweiterung des Systems würde neue, rechtliche Schwierigkeiten mit sich bringen.

Granularität soll insofern ein intuitives Verständnis dafür vermitteln, was geschieht, wenn die Anforderungen an Erfindungshöhe, Neuheit, Technizität und Präzision der Formulierungen der Patentansprüche sinken, wie in der Praxis seit Jahrzehnten zu beobachten. Amazons berüchtigtes "1−Click"−Patent ist ein Beispiel dafür.

Anders gesagt: ‘Software−Patente’ sind billig zu produzieren. Nicht mehr das Neue und Besondere, das Außergewöhnliche wird patentiert, sondern zunehmend das Alltägliche. Die Patentansprüche müssen nicht mehr exakt den zu patentierenden Gegenstand ("Erfindung") beschreiben, sondern können breit gefaßt werden und somit ganze ‘Klassen’ von Problemlösungen umfassen. Und schließlich werden ursprünglich dem Patentschutz nicht zugängliche Gegenstände durch Verbalakrobatik der ‘Stakeholder’, d.h. der Patentanmelder, Patentanwälte, Patentämter und Gerichte, patentierbar (Schölch 2001).

Wie in Abbildung 3 gezeigt, geht mit abnehmender Granularität eine zunehmende Anzahl von Patenten einher. Das Risiko, daß eines oder mehrere Patente den Lösungsbereich innerhalb einer bestimmten Problemdomäne tangieren oder durchsetzen, steigt analog. Jedwede Recherche stößt dort an Grenzen, wo die Patentanmeldungen noch nicht offengelegt worden sind. Hinzu kommt, daß ‘Software−Patente’ nicht etwa "als solche" angemeldet werden,[21] sondern beim Patentamt allen möglichen Patentklassen zugeordnet sein können.

---

[21] Diesbezüglich könnte die Umsetzung der geplanten EU−Richtlinie zu Software−Patenten Klarstellung bringen. Ob damit eine Verbesserung der Situation insgesamt einher gehen wird, ist gegenwärtig nicht abzusehen.

Bei dem beschriebenen exponentiellem Wachstum der Patentanmeldungen kann heutzutage nicht mehr davon ausgegangen werden, daß nennenswerte Bereiche der Softwareentwicklung nicht betroffen wären. Faktisch ist man inzwischen dort angelangt, daß ‘alles von Menschenhand Geschaffene unter der Sonne’[22] patentierbar ist, sofern es industriell brauchbar ist, d.h. entsprechende Wirtschaftsinteressen dahinterstehen.

Weniger offensichtlich ist eine andere Gefahr, die in Zukunft Europa erfassen wird: Patente für sogenannte ‘Computerprogrammprodukte’, wie sie etwa IBM in Gerichtsprozessen seit Jahren anstrebt. Hinter dem Wortungetüm verbirgt sich eine erhebliche Verschiebung im Machtgefüge zwischen Patenthaltern und Softwaredistributoren zugunsten der ersteren, die Patente als "Waffe gegen Kokurrenten" (Peters 2000) einsetzen.

Schlechte Aussichten für Europas Mittelständler: Patente auf ‘Computerprogrammprodukte’

Kleine und mittlere Unternehmen mit schwachem, oder nicht vorhandenem Patentportfolio können sich nur schlecht gegen Patentverletzungsklagen verteidigen. Zu unterscheiden sind zwei Typen von Verletzungsklagen, mit je unterschiedlich gravierenden Konsequenzen: Klagen wegen mittelbarer und unmittelbarer Patentverletzung.

> "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."[23]

Anders stellt sich die Situation bei einer mittelbaren Patentverletzung dar, wo eine solche "Einheit, bestehend aus Hardware und Software" nicht präsent ist. Normalerweise kann jemand, der eine Software, deren Funktion (zum Teil) patentiert war, ohne zugehörigen Computer vertrieb, höchstens wegen mittelbarer Patentverletzung vom Patenthalter verklagt werden. Ein Programm auf einer

---

[22] So der U.S. Supreme Court in Diamond v. Chakrabarty, 447 U.S.303 (1980).

[23] Lutterbeck, Horns & Gehring, 2000, S. 97.

Diskette ist nun mal ‘als solches’ nicht lauffähig ist, denn die patentierte Funktionsweise kommt ja erst durch das Zusammenspiel von Programm (auf Diskette) und PC zustande.[24] Erst wenn der Endanwender die Software in Betrieb nahm und dabei die Exklusivrechte des Patentinhabers ohne Lizenz oder gesetzliche Ausnahmebestimmung (etwa für Forschungszwecke) verletzte, konnte er wegen unmittelbarer Patentverletzung verklagt werden.

> "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’."[25]

Bei mittelbarer Patentverletzung ist die Beweisführung für den Patentinhaber erheblich schwieriger als bei unmittelbarer Patentverletzung und die vollständige Verhinderung des Vertriebs der Software quasi unmöglich, weshalb Patentinhaber es gerne sähen, wenn schon der bloße Vertrieb der Software durch Patente auf Software, die auf Datenträgern beliebiger Natur gespeichert ist, unter exklusive Kontrolle gebracht werden könnte. In den USA ist das Patentamt diesem Wunsch schon vor fast zehn Jahren gefolgt.[26]

Die Konsequenzen von Patenten auf ‘Computerprogrammprodukte’ dürften in Europa, wo die Softwareentwicklung wesentlich in kleinen und mittleren Unternehmen stattfindet, verheerend ausfallen. Dann wäre nämlich jeder Teilnehmer an der Softwareentwicklungs- und --distributionskette von Patentklagen bedroht, kommt es doch nicht auf das Wissen um eine etwaige Patentverletzung an. Potentielle Vertragspartner müßten dann immer darauf bestehen, daß ihre

---

[24] Selbst die Kombination aus PC−Hardware und Programm auf Diskette wird regelmäßig nicht ausreichend sein, um eine unmittelbare Patentverletzung herbeizuführen. Um das auf der Diskette gespeicherte Programm zu Ausführung zu bringen, wird normalerweise immer noch ein auf dem PC installiertes und lauffähiges Betriebssystem benötigt, das zudem noch mit dem Programm auf Diskette kompatibel sein muß. Nur im Zusammenspiel aus Hardware, Betriebssystem und Software (auf externem Datenspeicher), und ggf. weiteren Komponenten wie Gerätetreibern, läßt sich die patentierte Funktionalität überhaupt ‘herstellen’. Bei der juristischen Diskussion um das Für und Wider von Patentschutz für "Computerprogrammprodukte" wird gerade dieser Umstand häufig übersehen.

[25] Lutterbeck, Horns & Gehring, 2000, S. 99. Vgl. PatG §10.

[26] In Europa haben sich die Gerichte bisher erstaunlich widerständig gezeigt. Wie lange sie das nach Verabschiedung der zu erwartenden EU−Softwarepatentrichtlinie noch durchhalten, bleibt abzuwarten.

Lieferanten sie gegen Patentverletzungsklagen schadlos hielten.[27] Große Unternehmen mit eigenem Patentportfolio und eigener Patentabteilung werden dazu in der Lage sein, wie man es in den USA beobachten kann. Der typische deutsche und europäische Mittelständler dürfte dagegen hart getroffen werden, wenn er sich nicht dazu in der Lage sieht, und in der Folge seine Geschäftsbeziehungen wegbrechen.

Mit diesen Einsichten müssen Softwareentwickler wohl oder (eher) übel leben. Wie sie das tun können, wird in den nächsten Abschnitten diskutiert.

Ausflüchte und Auswege

Eine grundsätzliche Lösung der genannten −und anderer, nicht genannter− Probleme würde wohl nur eine tiefgreifende Reform des Patentwesens leisten können. Oder besser noch, es wäre ein eigenes Schutzrecht für Software zu schaffen, das die Spezifika ihrer Entwicklung und ihres Einsatzes angemessener berücksichtigt als Urheberrecht und Patentrecht heute leisten können. Eine solche Lösung ist jedoch nicht in Sicht und man darf mit einiger Berechtigung mutmaßen, daß die massiven Interessen der vom bestehenden Patentrecht profitierenden Stakeholder sie durch intensiven Lobbyismus zu verhindern wissen würden. Softwareentwickler die hier und heute mit dem Problem leben müssen, sind auf pragmatische Auswege und Ausflüchte angewiesen. Einige Ansätze werden im folgenden diskutiert.

Ignoranz

Ignoranz ist hier wortwörtlich zu verstehen: Der Softwareentwickler kümmert sich schlicht und einfach nicht um bestehende oder zukünftige Patente. Stattdessen entwickelt er seine Software so, wie er/sie es als Fachmann/-frau gelernt hat: Ausgehend von der Problemananlyse wird die Problämdomäne untersucht, eine Lösungsidee entwickelt und anschließend so implementiert wie es naheliegend ist, ohne Rücksicht auf Patente.

Ob ein solches Vorgehen sinnvoll ist, hängt von der individuellen Risikobewertung ab. Ist das Risiko, vom Patentinhaber belangt zu werden, gering, kann

---

[27] Sog. "idemnification and hold harmless agreements" (Stern, 1995, S. 77).

Ignoranz der Weg zum Erfolg sein, andernfalls sollte man andere Möglichkeiten (s.u.) in Betracht ziehen. In vielen Fällen wird das Risiko in der Tat sehr gering sein, wie sich zeigen läßt.

Angenommen, ein Unternehmen A vergibt an Softwareentwickler B, ein kleines Unternehmen mit 6 Mitarbeiter/inn/en, einen Auftrag zur Erstellung eines Warenwirtschaftssystems. Dasselbe soll über das Intranet mit einem Webbrowser zu benutzen sein. Wer jetzt mit der Patentlage im Bereich Internet−basierter Geschäftssysteme ein wenig vertraut ist, wird wissen, daß es sich um ‘Patent−vermintes’ Gelände handelt. Amazons "1−Click"−Patent und seine Abkömmlinge seien nur noch einmal als Beispiel erwähnt.

Intranet−Technologie unterscheidet sich im Kern nicht von Internet−Technologie, viele der diesbezüglichen Patente dürften also auch im Intranet Geltung haben. In Anbetracht der geringen Größe von B und der Kosten und des Zeitaufwandes für eine Patentrecherche mit ggf. zweifelhaftem Resultat, kommt eine solche für B ebensowenig in Frage, wie langwierige Lizenzverhandlungen mit einem oder mehr Patentinhabern.

Nun gibt es drei mögliche Fallkonstellationen, die für unsere Betrachtung von Interesse sind. (1) A hält die benötigten Patente und benötigt also keine Lizenz; (2) B hält die Patente und benötigt keine Lizenz; (3) Dritte (C) halten die Patente, d.h. A und B würden eine Lizenz benötigen.

Die Fallkonstellationen (1) und (2) sind uninteressant, interessant ist der Fall (3). Wenn sich A und B rational verhalten, werden sie stillschweigend auf die Einholung der Lizenz verzichten und zum gegenseitigen Vorteil das Geschäft mit der Softwareentwicklung abwickeln. Da das Programm im Intranet eingesetzt wird, werden Dritte von der unlizenzierten Nutzung ihres ‘geistigen Eigentums’ in der Regel gar nichts mitbekommen können. Und wo kein Kläger...

Ignoranz funktioniert also immer dann, wenn die ‘Kosten’ unbemerkt auf Dritte abgewälzt werden können, was in der Praxis häufig der Fall sein dürfte. Solange weder A noch B dazu übergehen, die Software an Dritte weiterzugeben, fällt

die Risikobewertung eindeutig zugunsten der Ignoranz aus.[28]

Cross−Lizenzen und Patent−Pools

In Industriebereichen mit blühemdem Patentwesen ist die Vermarktung von Innovationen ohne vorherige Beschaffung von Lizenzen äußerst schwierig, wenn nicht gar undurchführbar. Die im Markt agierenden Unternehmen haben daher zwei Grundinteressen. Zum einen möchten sie das benötigte, patentgeschützte Knowhow lizenzieren, um neue Produkte vertreiben zu können. Zum anderen wollen sie neu in den Markt drängenden Konkurrenten keine Marktanteile überlassen. In diesem Spannungsfeld sind Cross−Lizenzen und Patent−Pools angesiedelt.

Cross−Lizenzen sind gegenseitige Abkommen zweier Unternehmen, in denen die Bedingungen für die Nutzung der Patente −ausgewählter oder aller− aus dem Patentportfolio des jeweils anderen Unternehmens ausgehandelt sind. Charakteristisch ist, daß die Vertragsbedingungen solange beliebig ausgestaltet werden können, wie keine Wettbewerbsbehinderung entsteht. Je nach Verhandlungsposition der beiden Unternehmen werden die Vertragsbedingungen mehr oder minder symmetrisch ausfallen, was Umfang und Kosten der Lizenzierung angeht.

Patent−Pools stellen eine Erweiterung dieses Modells dar, bei der mehrere Unternehmen Schlüsselpatente halten, die zur Anwendung einer bestimmten Technologie unverzichtbar sind. In einem solchen Falle ist es für die Unternehmen kostengünstiger statt wechselseitiger individuelle Verhandlungen mit allen Unternehmen führen zu müssen, einen Patent−Pool einzurichten. Alle Beteiligten stellen dazu ihre relevanten Patente unter einer gemeinsamen Lizenz allen anderen zur Verfügung, individuelle Sonderkonditionen kommen nicht in Frage.[29]

---

[28] Ähnlich ist der Fall der ‘In−house’−Softwareentwicklung gelagert, nur daß es hier zwei statt dreier Interessenparteien gibt: Patenthalter und Softwareentwickler (der zugleich Anwender ist).

[29] Wettbewerbsrechtliche Argumente sprechen gegen individuelle Sonderkonditionen, da mit solchen eine allfällige Freistellung gemäß Art. 81 EGV gefährdet würde.

In beiden Fällen hängt die Legitimität der geschlossenen Verträge davon ab, daß der Wettbewerb nicht ungebührlich behindert wird, denn schließlich muß die Ausübung der gewerblichen Schutzrechte "im Einklang mit den Grundsätzen des lauteren Wettbewerbs stehen"[30]. In gewisser Hinsicht scheint das absurd, stellen doch Patente −als exklusive Rechte mit Verbotsanspruch− per se ein Wettbewerbshindernis erster Güte dar, das freilich gesetzlich legitimiert ist.[31] Warum sollte man an einen Patent−Pool höhere Wettbewerbsanforderungen stellen als an Patente als solche?

Eine klare Antwort gibt es auf diese Frage bisher nicht. Wie jedoch Shapiro[32] zurecht ausführt, stehen viele Unternehmen vor dem Problem, daß sie bestimmte Produkte nur dann auf den Markt bringen können, wenn sie dazu Lizenzen von anderen Unternehmen mit Patenten für komplementäre Technologien erhalten. Je größer die Anzahl der Patente und Patenthalter −wiederum eine Frage der Granularität−, desto drängender das Problem. Cross−Lizenzen oder Patentpools sind dann denkbare, wenn auch symptomatisch ansetzende, Gegenmittel gegen die durch das Patentwesen bewirkten Martkverwerfungen.

Die Crux beider Ansätze ist darin zu sehen, daß Unternehmen ohne eigene Patente in der Regel ausgeschlossen sind. Ohne Patentschutz steht zwar die von ihnen entwickelte Technologie den Patentbesitzern lizenzfrei zur Verfügung, umgekehrt ist das jedoch nicht der Fall. An dieser Stelle wirkt die wettbewerbsbehindernde Funktion von Patenten weiterhin fort. Vielen Unternehmen bleibt deswegen nichts anderes übrig, als sich ihrerseits mit Patenten zu ‘bewaffnen’, wollen sie nicht in absehbarer Zeit aus dem Markt gedrängt werden.

SAP, die lange ohne ‘Software−Patente’ auskamen −mit großem Erfolg, wie man betonen muß−, sind mittlerweile dazu übergegangen, Patente anzumelden.[33] So können auch sie in Zukunft bei Bedarf im Patent−Pool

---

30 Baumbach/Hefermehl, 1999, S. 101, RZ. 100.

31 Baumbach/Hefermehl, a.a.O.

32 Shapiro, 2000.

33 Für die Entscheidung ausschlaggebend war das aggressive Vorgehen von Konkurrenten insbesondere in den USA, die zunehmend Patente als "intelligente Waffen" (Rivette & Kline, 2000, S. 40) einsetzen. (persönliches Gespräch mit SAP−Mitarbeitern)

‘plantschen’ und müssen nicht zusehen, wie sie mit Hilfe staatlich verliehener Monopolrechte um wohlverdiente Marktanteile gebracht werden.

Davon hat allerdings die Open Source/Free Software−Community auch nichts, schließt doch ihr offenes Entwicklungsmodell die Anmeldung von Patenten weitestgehend aus.[34] Ihr Entwicklungs- und Vertriebsmodell, d.h. Software im Internet lizenzkostenfrei für Nutzung und Weiterentwicklung zur Verfügung zu stellen, wird insbesondere durch Patente auf Computerprogrammprodukte gefährdet, mit deren Hilfe der Vertrieb von Software allein schon als unmittelbare Patentverletzung unterbunden werden könnte (s.o.). Aber das wird die großen Patentanmelder[35] nicht unbedingt stören.

Liberal Licensing

IBM, als ein Unternehmen mit sowohl einem umfangreichen Patentportfolio als auch einer aufwendigen Open Source−Strategie, geht neue Wege, wenn es um die friedliche Koexistenz von Software−Patenten und Open Source−/Free Software geht. Im Rahmen des Eclipse−Projekts zur Erstellung einer Open Source−Entwicklungsumgebung hat IBM einen neuen Lizenztyp eingeführt, der hier als ‘liberal licensing’ bezeichnet werden soll, und der einige Ähnlichkeit mit dem Ansatz für ‘Open Software Patents’[36] aufweist (s.u.).

Der Grundgedanke der Eclipse−Lizenz besteht darin, daß alle Projektteilnehmer sich wechselseitig verpflichten, bestehende Patentansprüche nicht gegen die entwickelte Software durchzusetzen:

> "Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non−exclusive, worldwide, royaltyfree patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder."

---

[34] Lutterbeck, Horns & Gehring, 2000, S. 10 f.

[35] Laut Scheeres, 2003, hat Microsoft bereits mehre hundert und IBM mehr als 3000 Software−Patente beim Europäischen Patentamt in München angemeldet.

[36] Gehring, 2000.

Im Unterschied zur GPL[37] und anderen Free Software/Open Source−Lizenzen, die urheberrechtlich wirken, erstreckt sich diese Freistellungsklausel nicht auf den Code, sondern auf das Programm. In der Folge ließen sich keine Programmteile zu anderen Zwecken lizenzfrei widerverwenden.

Das Open Source−Software−Patente−Dilemma wird wiederum nur symptomatisch und nicht grundsätzlich überwunden. Inwieweit dieses Beispiel Schule machen wird, bleibt abzuwarten.

Dual Licensing

Dual Licensing ist eher aus dem Bereich der urheberrechtlichen Lizenzierung bekannt denn von Patentlizenzen. Beim Dual Licensing macht der Rechteinhaber dahingehend von seinen Exklusivrechten Gebrauch, daß er Nutzungsrechte in Abhängigkeit von Zweck und Gestalt der Nutzung kostenpflichtig oder kostenfrei gestattet.

Das wohl prominenteste Beispiel für Dual Licensing im Bereich der Software−Patente ist die "RTLinux Open Patent License, version 2.0" der Firma FSMLabs.[38] Die genannte Lizenz bezieht sich auf das U.S.−Patent 5,995,745, das dem FSMLabs−Entwickler Victor Yodaiken am 30. November 1999 zugesprochen wurde.[39]

Gemäß der "RTLinux Open Patent License, version 2.0" darf die patentierte Erfindung unter bestimmten Bedingungen, d.h. insbesondere im Zusammenhang mit der Entwicklung freier Software, ohne Zahlung von Lizenzkosten genutzt werden:

> "In addition to the other terms and conditions of this License, use of the Patented Process is permitted, without fee or royalty, when used:
>
> A. By software licensed under the GPL; or

---

[37] Zu Inhalt und Anwendung der GPL siehe insbesondere die umfangreichen Ausführungen unter http://www.gnu.org/licenses/gpl-faq.html.

[38] Der Text der Lizenz ist im Internet einzusehen unter http://www.fsmlabs.com/products/rtlinuxpro/rtlinux_patent.html [01 Sep 2003].

[39] Das Patent, ein klassisches Software−Patent, deckt bestimmte Modifikationen von Standard−Betriebssystem ab, um diese mit ‘real time’−Fähigkeiten auszustatten, wie sie beispielsweise für den Einsatz in Steuerungssystemen gefordert werden.

B. By software that executes within an Open RTLinux Execution Environment − whether that software is licensed under the GPL or not."

Für alle anderen Nutzungsformen entfällt die ‘automatische’ Freistellung von Lizenzzahlungen.

Aus Sicht der Open Source−/Free Software−Entwickler ist eine solche Lizenz natürlich ein Schritt in die richtige Richtung. Da es aber grundsätzlich ins Ermessen des Lizenzgebers gestellt ist, die Lizenzbedingungen für zukünftige Nutzungen zu ändern, ist wirkliche Rechtssicherheit nicht gegeben. Auch Entwicklern proprietärer Software ist mit einer solchen Lizenz nicht geholfen, wollen sie ihre Software doch gerade nicht ‘frei’ (im Sinne der GPL) vertreiben.

Der "Berliner Ansatz zu ‘Open Software Patents’" (Gehring 2000)

Wie im vorletzten Abschnitt bereits erwähnt, gibt es Ähnlichkeiten zwischen dem "Berliner Ansatz zu ‘Open Software Patents’" und dem ‘Liberal Licensing’. Allerdings ist ersterer deutlich breiterer konzipiert worden. In Anlehnung an die ‘Vererbung’ der Lizenzrechte und −pflichten, wie sie aus der GPL bekannt ist, werden Lizenznehmer zur lizenzkostenfreien Zurverfügungstellung von Nutzungsrechten aus Patenten verpflichtet (so vorhanden), wenn sie selbst Urheberrechte und/oder Patentrechte an der betroffenen Software in Anspruch nehmen wollen. Eine Einschränkung auf ein einzelnes Programm ist nicht vorgesehen, auch Programmbestandteile wären abgedeckt. Man kann sich das Ergebnis als universellen Patent− und Urheberrechtspool vorstellen, innerhalb dessen ein ‘Waffenstillstandsabkommen’ gilt. Je mehr Teilnehmer dieser Pool hat, desto weiter reicht der ‘allgemeine Landfrieden’.

Natürlich werden nicht alle Unternehmen von einer solchen Idee begeistert sein. Flankierend sollte Free− und Open Source Software−Entwicklern deshalb durch eine Modifikation des Patentrechts (Neuheitsschonfrist) die Möglichkeit an die Hand gegeben werden, ihrerseits Patente als ‘Schild’ gegen die Patente von aggressiven Konkurrenten zu erwerben, wie sie den Entwicklern proprietärer Software zur Verfügung steht.[40]

---

[40] Diese Forderung wurde von Lutterbeck, Horns & Gehring, 2000, S. 10, erneut aufgegriffen.

Wie alle vorherigen Ansätze, würden Open Software Patents lediglich schwere Auswüchse des Patentwesens bekämpfen können. Auf lange Sicht könnte es für die beteiligten Entwickler und Unternehmen jedoch sinnvoll sein, auf diese Weise die Kosten für Softwareentwicklung und −distribution, sowie für die Beilegung von Streitfällen, zu verringern.

Ein letzter Ansatz bleibt vorzustellen, der eine auf die Entwicklung quelloffener Software zugeschnittene Modifikation des Patentrechts vorschlägt.

Das Quelltextprivileg (Lutterbeck, Horns & Gehring 2000)

In einem Gutachten für das Bundeswirtschaftsministerium untersuchten Lutterbeck, Horns & Gehring (2000) die Auswirkungen des Patentschutzes für "computer−implementierte Erfindungen" auf die Entwicklung von quelloffener Software im allgemeinen und mögliche Implikationen für die Sicherheit der Informationstechnologie im besonderen.

Die Autoren stellen fest, daß kleine und mittlere Unternehmen ebenso wie die Entwickler quelloffener Software durch das bestehende Patentsystem benachteiligt werden. Dadurch wird, insbesondere in Deutschland und Europa, das bestehende Innovationspotential erheblich bedroht. Hinzu kommt, daß sicherheitsrelevante Softwaretechnologien sich nicht so weit verbreiten, wie es im öffentlichen Interesse wünschenswert wäre. In der Schlußfolgerung wird die Einführung eines Quelltextprivilegs empfohlen:

> "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)"[41]

Das Quelltextprivileg, wie vorgeschlagen, würde den Softwareentwickler zu einem Teil von Klagedrohungen wegen mittelbarer und/oder unmittelbarer Patentverletzung entlasten. Die oben geschilderte Gefahr der Klage wegen Patenten auf Computerprogrammprodukte könnte gebannt werden, indem

---

[41] Lutterbeck, Horns & Gehring, 2000, S. 9.

Software im Quelltext vertrieben würde. Softwarevertrieb im Quelltext könnte wiederum einem Qualitäts− und Sicherheitswettbewerb förderlich sein.[42]

Ergänzend schlagen die Autoren die Etablierung einer öffentlichen Zeitstempelstelle für Softwarequelltexte vor, um bei etwaigen Streitfällen Rechtssicherheit zu schaffen.[43] Durch eine solche Anlaufstelle könnte das eingangs erwähnte Problem der Recherche des Standes der Technik partiell gelöst werden, indem sich die Suche nach einer bestimmten Technologie eingrenzen ließ. In Kombination mit einer einheitlichen Softwarebeschreibungssprache (z.B. UML) ließe sich im Laufe ein wertvolles Recherchewerkzeug etablieren.

Aber auch bei diesem Vorschlag handelt es sich um eine Symptombehandlung, die allenfalls einige, wenige Schwächen des Patentsystems lindern hilft. Gesund wird der Patient dadurch noch lange nicht.

Fazit

Bis in die späten sechziger Jahre konnte nach allgemeiner Auffassung, die von den seinerzeit führenden Lehrbüchern geteilt wurde,[44] Software nicht patentiert werden. Dafür hatte sich insbesondere die Computerindustrie mit ihrem damaligen Weltmarktführer IBM stark gemacht. In einem Artikel der New York Times mit der Überschrift "Software is unpatentable" wird ein Sprecher der IBM mit den folgenden Worten zitiert:

"The primary reason for our concern in the patenting question is that our customers would face many problems if programs were patentable. One of those problems would be not knowing whether the programs they are writing are free from infringements from valid patents."[45]

Software wurde seinerzeit in der Hauptsache gebündelt mit Hardware geliefert, im Quelltext, um den Kunden die Gelegenheit zu geben, den Code ihren Bedürfnissen entsprechend anzupassen. IBM mußte daher befürchten, daß ihr

---

[42] Ausführlicher zu diesem Aspekt Gehring, 2003.

[43] Lutterbeck, Horns & Gehring, 2000, S. 11.

[44] Tapper, 1983, S. 2, Fn 4; Bender, 1988, Abschnitt 3.02(4), S. 3−9.

[45] Jones, 1968.

Hauptgeschäft, der Verkauf von Hardware, durch ‘Software−Patente’ beeinträchtigt würde, wenn die Kunden fürchten müßten, wegen unabsichtlicher Patentverletzungen belangt zu werden.

In der Zwischenzeit haben sich die Vermarktungsstrategien für Software verändert, aber die Prinzipien der Softwareentwicklung sind gleich geblieben. Das Prinzip der Quelloffenheit von Software, das seinerzeit der Normalfall war, ist in dieser Zeit dabei, Märkte neu zu besetzen. Die Entwickler stoßen auf die gleichen Schwierigkeiten, die seinerzeit Wissenschaft und Praxis zu den Warnungen vor Software−Patenten veranlaßt hatten.

Die frühen Warnungen sind ebenso überhört worden wie die aktuellen, wie die New York Times 30 Jahre später in einem Artikel feststellen muß:

> "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."[46]

Alle noch so intelligenten Konstruktionen führen nicht an einer ernüchternden Einsicht vorbei: Eine Patentlösung für die durch ‘Software−Patente’ verursachten Schwierigkeiten bei der Softwareentwicklung ist derzeit nicht in Sicht.

---

[46] Gleick, 2000.

Literatur

Baumbach, A. & Hefermehl, W.: Wettbewerbsrecht, 21. Aufl., C.H. Beck, München 1999.

Bender, D.: Computer Law. Software Protection, Matthew Bender, New York 1988

Bowonder, B. & Yadav, S.: R&D pending patterns of global firms. Research Technology Management, November/Dezember 1999, S. 44−55.

Bragg, A. W.: Software patents , IEEE IT Professional 2001 3(4), S. 39−42.

Challet, D. & Du, Y. L.: Closed source versus open source in a model of software bug dynamics. Preprint , 2003.
*http://arxiv.org/abs/cond-mat/0306511 [29 Aug 2003].

Cohen, J. & Lemley, M.: Patent Scope and Innovation in the Software Industry, 89 California Law Review 1, 2001.

Gehring, R. A.: Der Berliner Ansatz zu Open Software Patents. Beitrag zur Konferenz über wirtschaftspolitische Aspekte der Patentierung von Software, Bundeswirtschaftsministerium, Berlin, 18. Mai 2000.
*http://ig.cs.tu−berlin.de/ap/rg/2000-05/Gehring-OpenSoftwarePatents−052000.pdf [24 Aug 2003]

Gehring, R. A.: Software development, intellectual property rights, and IT security , The Journal of Information, Law & Technology (JILT), 8(1), 2003.
*http://elj.warwick.ac.uk/jilt/03-1/gehring.html [21 Aug 2003]

Gleick, J.: Patently absurd, New York Times Magazine v. 12.3.2000

Grenzmann, C. & Greif, S.: Relationship between R&D input and output, in OECD, ed., Innovation, Patents and Technological Strategies , OECD, Paris 1996, S. 71−88.

Heuer, S.: Boston Gene Party, in Brand Eins 02/2003, S. 20−28.

Himmelein, G.: Softwarepatent: Adobe verklagt Macromedia , heise newsticker (11. 8.2000).
*http://www.heise.de/newsticker/data/ghi-11.08.00-000/ [25 Aug 2003]

Homann, L.: Beyond Software Architecture. Creating and Sustaining Winning Solutions, Addison-Wesley, Boston u.a. 2003.

Horns, A. H.: Anmerkungen zu begrifflichen Fragen des Softwareschutzes, in GRUR 2001, 1−16.

Jones, C.V.: Software is unpatentable, New York Times v. 23.10.1968, S. 59.

Johnson, L.: Creating the software industry , IEEE Annals of the History of Computing, 2002, 24(1), S. 14−42.

Kiesewetter−Köbinger, S.: Über die Patentprüfung von Programmen für Datenverarbeitungsanlagen (2000).
*http://swpat.ffii.org/papers/grur-skk01/index.de.html [12 Aug 2003].

Koboldt, C.: Much pain for little gain: A critical view of software patents, The Journal of Information, Law & Technology (JILT), 8(1), 2003.
*http://elj.warwick.ac.uk/jilt/03-1/koboldt.html [21 Aug 2003].

Kovitz, B. L.: Practical Software Requirements, Manning, Greenwich, CT, 1999.

Kretschmer, M.: Software as text and machine: The legal capture of digital innovation The Journal of Information, Law and Technology (JILT), 8(1), 2003 (1). *http://elj.warwick.ac.uk/jilt/03-1/kretschmer.html [11 Aug 2003]

Lutterbeck, B., Horns, A. H. & Gehring, R. A.: Sicherheit in der Informationstechnologie und Patentschutz für Softwareprodukte − ein Widerspruch? [Security in information technology and patent protection for software products: A contradiction?] Berlin 2000, *http://ig.cs.tu-berlin.de/forschung/IPR/LutterbeckHornsGehring-KurzgutachtenSoftwarePatente-122000.pdf [28 Aug 2001]

OECD: Innovation, Patents and Technological Strategies, Paris 1996.

Peters, M.: Patente werden immmer wichtiger. Interview mit Rolf Ohmke, Leiter der für Kommunikationstechnik zuständigen Patentabteilung bei Siemens/München, in Tagesspiegel v. 7. Mai 2000, S. 23.

Rivette, K. G. & Kline, D.: Wie sich aus Patenten mehr herausholen lässt, in Harvard Business Manager, 4/2000, S. 28−40.

Samuelson, P. & Scotchmer, S.: The Law and Economics of Reverse Engineering, 111 Yale Law Journal 1575, 2002.

Scheeres, J.: European Patent Law Draws Fire, Wired News vom 1. September 2003. *http://www.wired.com/news/technology/0,1282,60245,00.html [01 Sep 2003].

Schölch, G.: Softwarepatente ohne Grenzen, GRUR 2001, S.16−21

Shapiro, C.: Navigating the patent thicket: Cross licenses, patent pools, and standard−setting. Competition Policy Center. Paper CPC00−011, 2000.

Stern, R. H.: Micro law. Software patents, IEEE Micro, April 1990, S. 8−11.

Stern, R. H.: Micro law. The PTO on software patents, IEEE Micro, August 1995, S. 2−3, 77−78.

Stock, M.: Rechtsschutz für Software: Urheberrecht oder Patentrecht? Ein Schutz mit sieben Siegeln?! , PASSWORD, (07 & 08), 2001, S. 20−28.

Tapper, C.: Computer law, 3. Aufl., Longman, London and New York 1983.

van Zuylen, H. J.: The Redo Compendium. Reverse Engineering for Software Maintenance, John Wiley & Sons, New York u.a. 1993.