Journal of Information, Law and Technology

Software as Text and Machine: The Legal Capture of Digital Innovation

Martin Kretschmer
Centre for Intellectual Property
Policy and Management
Bournemouth University

This is an **editorial** published on: 4 July 2003

Citation: Kretschmer M, Software as Text and Machine: The Legal Capture of Digital Innovation 2003 (1) *The Journal of Information, Law and Technology (JILT)*. <http://elj.warwick.ac.uk/jilt/03-1/kretschmer.html>

1. Introduction

The wave of digitisation associated with the microchip (INTEL, 1971), the personal computer (MITS Altair 1975, APPLE II, 1977; IBM PC, 1981) and the Internet (TCP/IP communication protocol, Defense Advanced Research Projects Agency DARPA, mid-1970s; WorldWideWeb, European Organization for Nuclear Research CERN, 1989) arrived without warning, and still shows little sign of petering out. Digital representation and functionality now pervade most processes and products of modern economies. Yet concepts of intellectual property were unprepared for their reception. Algorithms of abstract beauty suddenly had become part of 'machines', with large scale industrial consequences. Threshold 'inventive steps' disappeared in a widespread innovative turmoil. In governing networked communications and transactions, software itself acquired normative force.

In 1972, the US Supreme Court ruled in Gottschalk v. Benson (409 U.S. 63) that computer programs were not patentable, drawing on a 1966 Presidential Commission on the Patent System. When the European Patent Convention was signed in 1973, its exclusions on patentability under Art. 52 (including programs for computers 'as such') reflected a widespread consensus. Yet by 1999, the annual number of software patents granted in the US had risen to about 20,000. The European Commission estimated that 'although the Munich Convention and the national laws of Member States do not permit the patentability of computer programs as such, there are about 13,000 European patents covering software!'

Why did advanced intellectual property systems experience this dramatic about-turn? What are the economic and societal implications of a shift towards exclusive rights on the functionality rather than the code of software? Are there good reasons to reject the doctrine of software copyright in favour of the patentability of computer programs? Can these competing modes of protection co-exist? These are pressing questions on the eve of a European Directive on the Patentability of Computer-Implemented Innovations which seeks to legitimate and perpetuate current European Patent Office (EPO) practice, allowing patents on software if they are structured in a prescribed way showing a 'technical contribution' (a criterion developed following the 1987 EPO Board of Appeal decision in Vicom, T208/84).

This special issue assembles contributions from a symposium on 'software-related inventions' organised in June 2002 by the Centre for Intellectual Property Policy & Management (CIPPM) at Bournemouth University. The symposium sought to reflect broad experience about software development and current patenting practice, rather than focusing on legal doctrine or ideological divisions. It brought together IT specialists both from small companies and multinationals, legal masterminds in drafting software patents, and social scientists with a track record of empirical work in the area. The flexibility of electronic publishing under the umbrella of JILT facilitated both peer reviewed academic articles while also preserving the contributions from practitioners and industry, reflecting some of the refreshing spirit of the day.

Broadly speaking, smaller software developers and the open source movement reject software patents, sometimes with evangelical fervour. Economically trained academics share grave doubts about the virtues of software patenting in promoting innovation. Lawyers, by contrast, seem prepared to advance the art of the possible (in particular if a client is at hand).

Larger firms defend the incentive character of a patent system available to all areas of technology. They are also concerned about the predictability of the process, and the possibility of cross-licensing.

This introduction aims first to provide a sketch of the run-up to the present intractable stand-off between proponents and opponents of software patents. Secondly, it will introduce the three sub-themes around which this special issue is organised: current practice, economic analysis and prospects for a licensing economy.

2. Historical Excursion

Intellectual property law responded to digitisation through case decisions and doctrinal debate rather than fundamental policy review. The rationale for providing a particular type of software protection was rarely assessed from first principles. Much early discussion negotiating software's legal status involved exercises in probing and stretching concepts: Yes, software is more than a text: it does something. But so does musical notation? Why are piano rolls not deserving of patent protection? Can software be a work? Can a computer program be considered a pure intellectual act if it uses natural forces?

Copyright may have evolved as the international mechanism of choice because it could accommodate lobby interests with a simple conceptual redefinition. In 1978, a US Commission appointed by Congress (CONTU), supported software copyright. Within 16 years, Article 10 of TRIPS, the cornerstone intellectual property agreement of the WTO (1994), issued a blunt prescription to a global audience: computer programs in source or object code are 'literary works under the Berne Convention' What would Victor Hugo have made of that, as he and his disciples worked in the Association Littéraire et Artistique towards the Berne Convention of 1886?

It may be useful to retrace the steps towards the policy recommendation of protecting software under copyright, before examining the subsequent rise of software patenting. This is initially American legal history, because computer technology was first commercialised in the United States.

2.1 Software Pre-Benson (1972)

A computer program is a set of instructions intended to bring about certain results. It can be implemented in hardware or software. Until the mid-1960s, hardwiring was the prevalent way of transporting instructions. Computer programs are valuable not because they can be appreciated as texts -- in source code or binary (machine-executable) object code -- but because they make computers perform tasks. Thus software developers always have understood computers as machines, frequently resorting to metaphors from the industrial world. Buyers have always bought software because of its functionality (in combination with complementing hardware and system software). Patent concepts might have been applied, as in the 1966 UK decision allowing a claim to a computer 'when programmed to solve a linear programming problem by an iterative algorithm' since it may be regarded as 'a machine that has been temporarily modified'.

Johnson (2002, p.14) gives as good summary of the start of the software products industry:

> '[Today] it is almost inconceivable that only 40 years ago the concept of software as a commercial product was considered harebrained. Yet that was the case in the 1960s. Computer users had limited choices for acquiring the software they needed to run their applications. They could obtain generalized programs from their hardware vendor at no cost because the cost of software was bundled into the computer's cost. Their second choice was to create, at great expense by using their own programmers or a contract programming firm, customized programs designed to their own specifications. Software was either free, obtained from the computer manufacturer, or customized for use by a specific customer only. Consequently, it seemed an impossibility to design software generalized enough to be sold to multiple users yet differentiated enough from the hardware manufacturers' free software that customers would willingly pay for it. The 1960s were boom years for entrepreneurial firms established to sell programming and system design skills under contract in a market where the rapidly expanding use of computers created a high demand for those skills. The first of these companies - Computer Usage Corporation - was founded earlier, in 1955, but by the end of the 1960s, there were thousands of such firms. Most such companies were small, but a handful were large enough to go public, employing hundreds of programmers. These firms increasingly found opportunities to package the software they had already written and deliver it to multiple customers, a situation that promised potentially high profits given the low cost to reproduce already developed software. The term software packages appeared in the late 1960s and implied that the customer deliverables included documentation and some level of service, such as installation, as well as the program code. Many early products were utility programs with greater functionality or efficiency than the comparable free software from the hardware vendors. Other early products were software applications like payroll or banking where external factors such as government regulations imposed a uniformity on the way that customers defined their specifications. In January 1967, International Computer Programs (ICP) in Indianapolis, Indiana, began publishing a quarterly catalogue of computer programs available for sale, and the software product industry began to take shape.'

Software patentability was tested throughout the 1960s, with mixed results. The most important case concerned an application filed in 1963 by Bell Labs (on behalf of employees Benson and Talbot) for a method for converting from binary-code decimals to pure binary numbers. The US Patent Office examiner rejected the application because the claims did not recite statutory subject matter (being about 'mental processes' and 'mathematical steps'). This decision was overturned twice on appeal before reaching the Supreme Court. The reasoning of an unanimous court (Gottschalk, Commissioner of Patents v. Benson; 409 U.S. 63; 1972) has been criticised by many as impenetrable. The court characterised the process claim as abstract and sweeping, in effect a patent on the algorithm itself, yet explicitly did not preclude the possibility of software patents: 'We do not hold that no process claim could ever qualify' (409 U.S. at 71). A 1966 Presidential Commission on the Patent System is cited, which recommended that software patents should not be permitted, because satisfactory growth in

the industry had taken place in the absence of patent protection; and reliable prior art searches would not be available. The Benson court quotes the lines: 'Indirect attempts to obtain patents and avoid the rejection, by drafting claims as a process, or a machine or components thereof programmed in a given manner, rather than as a program itself, have confused the issue further and should not be permitted.' If there is a line to be drawn on the patentability of computer algorithms, it is certainly not clear where. Benson was generally taken to reject software as statutory subject matter.

The 1966 Commission on the Patent System had also argued against software patents because copyright protection was already available (a line also quoted by the Benson court). And indeed, the US Copyright Office did accept deposits of computer programs since 1961. Copies deposited were regarded as 'how to' books, and had to include code in a language intelligible to humans (i.e. source code).

2.2 CONTU (1978) and Japan's Sui Generis Proposal (MITI, Pre-1985)

Following the decision in Benson, software copyright was in the ascendancy. Its first statutory expression can be found in the 1976 US Copyright Act which provides for a wide definition of literary works, including 'works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia' (Section 101). Congressional intent made it clear that 'the expression adopted by the programmer is the copyrightable element in a computer program, and that actual processes or methods embodied in the program are not within the scope of the copyright law' (House Report, 94th Congress, 1476 [1975]). The precise scope of protection was left open in order not to delay the Act. The software ambit of the problematic idea-expression doctrine as well as special exceptions to the exclusive rights provided were delegated to a Commission on New Technology Uses of copyright works (CONTU, 1975-78). CONTU considered the demarcation lines between flow chart, source code and object code, all found to be 'protectable expressions'. A sui generis approach to software was rejected, leaving software with the long term and low threshold requirement of copyright law. Certain exceptions for back-up copies and adaptations 'essential' to the utilization of computer programs were introduced (new Section 117, adopted 1980). The CONTU report also gives priority to licensing terms (p. 13): 'Should proprietors feel strongly that they do not want rightful possessors of copies of their programs to prepare such adaptations, they of course, make such desires a contractual matter.'

A major factor in the subsequent development of international law was an industrial espionage case in the early 1980s involving IBM and Hitachi which produced increasingly successful IBM compatible computers. IBM set up a plot in order to trap affiliated US employees of Japanese clone makers into buying sensitive documents, including software developmental materials and manuals. In settlement agreements signed by Hitachi, and Fujitsu in 1983, the Japanese firms agreed to pay IBM fees even for software developed independently. All new products would be exposed to IBM's scrutiny for copyright infringing similarities. Hitachi said about the criminal case based on trade secret laws: 'IBM's aim from the start was to get us to recognize its copyright.'

The case was seen in Japan as a national disgrace. The powerful ministry for international trade and industry (MITI) set about devising a software law that would reduce the control of the earliest developer of whom subsequent developers may be said to be derivative. It proposed a sui generis approach, introducing a software registry as prerequisite of protection, limiting protection to 15 years, and including provisions for compulsory licences.

Under heavy US pressure (the European Commission filed a notice supporting the US position), MITI's sui generis proposal was abandoned in favour of a re-drafting of copyright law supported by the Cultural Affairs Agency. In 1985, the Japanese Copyright Act was amended with a definition of computer programs. CONTU's recommendations had become the international blueprint. Precedents from the Australian, German and French courts followed similar lines. A European Software Directive was eventually adopted in 1991 (91/250/EC). The NAFTA (1993) and TRIPS (1994) Agreements and the WIPO Copyright Treaty (1996) completed the process, classifying computer programs in source or object code as 'literary works'.

2.3 Copyright on Software Functions (Whelan 1986; Altai 1992)

The juridical legerdemain of software copyright proved initially quite benign. Software innovation was not captured by copyright, since competing software developers could offer similar solutions to similar problems without literally copying code. Copyright infringement evidence (rather than straight product piracy) was in many cases dependent on the mistakes of defecting employees who only insufficiently covered their tracks. However, over time it became clear that copyright law failed to address issues central to a digitised world. Software copyright may have facilitated distribution of software as binary (unintelligible) machine code, guarding the source code as trade secret, and preventing the interoperability of competing applications.

Subsequent policy developments were less concerned with these shortcomings than with the increasing strategic importance of intellectual property generally. The twin face of software as text and machine made it a natural locus for probing the scope of the rights provided. The first stream of decisions extending copyright protection beyond the literal elements of software again came from the US. In Whelan Associates v. Jaslow Dental Laboratory (3rd Circ.1986 - 797 F.2nd 1222), a program of similar architecture had been rewritten in a different programming language; 'five particularly important 'subroutines' within both programs -- order entry, invoicing, accounts receivable, end of day procedure, and end of month procedure -- performed almost identically in both programs' (p.1228). The appeals court deemed the 'structure, sequence and organization' of a particular program protectable if the desired purpose of the program could have been achieved by other means. The Whelan decision was criticized for extending far broader protection to computer techniques than would have been possible under patent law.

In Altai, another US appeal court modified the Whelan approach by introducing into software copyright an abstraction-filtration-comparison test: in effect 'a patent registration without patent claims' (Aharonian, 2001, p. 15).

In ascertaining substantial similarity under this approach, a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or possibly kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program. The result of this comparison will determine whether the protectable elements of the program at issue are substantially similar so as to warrant a finding of infringement. (Computer Associates International v. Altai, 2nd Cir. 1992 - 982 F.2d 693, at 1253)

The purpose of the program constitutes the highest level of abstraction -- and would be filtered out as unprotectable 'idea'; the source and object code constitute the lowest level of abstraction -- and copying is likely to be an infringement. The mid-level of operations and data types (e.g. algorithms and data structures) will generally be the most contentious, and it is less than clear that Altai's test can assist here without a series of precedents defining concepts of 'standard techniques' analogous to patent law's 'prior art', and 'functional constraints' analogous to patent law's 'obviousness'.

2.4 GPL (1988) and Open Source

The encroachment of software copyright onto patent territory was not a phenomenon limited to the US. Many jurisdictions experienced similar trends, extending through case decisions the scope of protection beyond literal code to structural elements of the program. In the academic software community that had pioneered the Internet, a belief grew that it was precisely accumulated knowledge about the mid-level functionality of programs that would improve the quality and normative acceptability of software. Copyright did not appear to make these elements sufficiently available to subsequent developers. As software increased in complexity (and higher level programming made use of obfuscation techniques), decompilation and reimplementation of a program from the binary object code (in which it is distributed and executed) became more difficult. In order to understand a program fully, access to the source code (including symbolic labels and annotations) was indispensable. Richard Stallman pioneered an open approach to software development and distribution in the GNU Project, launched in 1984 in order to develop a complete Unix-like operating system. In 1988, Stallman issued the first version of the General Public License (GPL) forcing derivatives of GNU software to keep their source code free from proprietary claims. In a radical spirit, which has been described as the constitution of the Free Software/Open Source movement, copyright law was used to subvert itself. The key terms of the GPL are the following (version 1991):

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; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

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:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. (…)

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.).
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable.

2.5 Software Patents Revived (Diehr 1981; Vicom 1987; State Street Bank 1998; Proposed EC Directive 2002)

In parallel to this battle for the soul of software copyright, software patents began to resurface, sometimes with paradoxically similar arguments to the open source movement. Is it not part of the rationale of the patent system to achieve disclosure of important techniques which would otherwise remain trade secrets? Would software patents not make software innovation more widely known?

During the 1970s, there was continuous low level patenting of software related inventions as long as drafting complied with the dictum that a computer program could not be the only novel feature of an invention. A few hundred such patents were issued by the USPTO each year, and

the Supreme Court in Parker v. Flook (437 U.S. 584; 1978) deflected another attempt to establish software as patentable subject matter. Rejecting a method claim to updating alarm limits in a known chemical process (the catalytic chemical conversion of hydrocarbons), the court held that '[t]he process itself, not merely the mathematical algorithm, must be new and useful.' (p. 591)

In Diamond v. Diehr (450 U.S. 175; 1981), the next -- and until now last -- software patent case before the US Supreme Court, the ruling was structured the other way around. A well-known mathematical formula, the Arrhenius equation, was used to calculate the cure time in a rubber-moulding press. The court viewed the claimed invention as a conventional process. 'Their process admittedly employs a well known mathematical equation, but they do not seek to pre-empt the use of that equation. Rather, they only seek to foreclose from others the use of that equation in conjunction with all the other steps in the claimed process.' (p. 185f; 5:4 majority decision)

After Diehr, patenting strategies shifted openly towards indirect drafting of software claims. The 'algorithm' exclusion to patentability could now be subverted by any kind of practical application, including a programmed general purpose computer where the novelty arises from the software itself (not a 'physical transformation'). The USPTO Guidelines of 1996 acknowledged the new practice:

> The subject matter sought to be patented must be a 'useful' process, machine, manufacture or composition of matter, i.e., it must have a practical application.... The utility of an invention must be within the 'technological' arts. A computer-related invention is within the technological arts. (61 Fed. Reg., p. 7480).

The European Patent Office had developed a similar approach following the Vicom decision of 1987 (EPO Board of Appeal, T208/84). The disputed patent was about a digital image enhancement technique claimed as a method of two-dimensional data convolution. The claims in the initial form were not specific to the processing of digital images but to a method of digital filtering, in effect a multipurpose algorithm. This had been rejected upon examination. Before the Board of Appeal, the applicants agreed to limit their claims to a method of digitally processing images. This was allowed because it specified 'what physical entity is represented by the data and forms the subject of a technical process, i.e. a process which is susceptible to industrial application'.

In subsequent EPO practice, this meant (according to Simon Davies' contribution below) 'that all inventions that might reasonably be considered as within the realms of computer science, for example procedures at the operating system level to improve machine operation, or generic algorithms, techniques and functionality at the application level' would normally be regarded as patentable.

The 1994 TRIPS Agreement provides in Article 27.1 that 'patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application'. In order to appear within the 'technological arts' under the USPTO 1996 Guidelines, software claims had to be

drafted to a machine (that machine being a general purpose computer) rather than to a process or method running on a general purpose computer. The 1998 decision in State Street v. Signature Financial Group (149 F3.rd 1368; Court of Appeal for the Federal Circuit) removed this hurdle by allowing claims to methods if they are not 'merely abstract ideas constituting disembodied concepts or truths that are not 'useful" (p. 1373). Following State Street, subject matter exclusions effectively disappeared from the US examination process. The number of annual US software related patents doubled from around 10,000 in 1996 to around 20,000 in 1999, including a significant number on software implemented business methods. This accelerated an international change in patenting policy leading to the current intervention by the European Commission.

In 1997, IBM constructed, apparently in collusion with the European Patent Office , two appeals to the EPO Board of Appeal claiming patentability of a computer program directly (not indirectly as a system or method). The inventions had been chosen as clearly patentable in line with previous EPO decisions, just the form of claims was drafted in order to challenge the Article 52 exclusion on computer programs 'as such' The inventions concerned a Windows display program that prevents simultaneously displayed windows obscuring each other (T935/97), and a program that automatically recovers from failure of a resource called upon by the program (T1173/97). The appeals were heard in 1998 and issued in March 1999. Sailing close to the wind, the Board of Appeal argued that a technical effect is present if the program enables an effect 'beyond the 'normal' physical interactions between the program (software) and the computer (hardware)'. In plainer words: software is not a computer program 'as such' if it is innovative and works.

The practical effects of being able to claim a computer product by itself or as a record on a carrier are not insignificant but limited. Competitors only infringe a method patent directly if they ran the program, not if they make and sell disks containing programs using the patented method. Thus infringement proceedings are more straightforward for software patents claimed as products. But in attempting to clarify and simplify the law, IBM had re-opened a fundamental policy debate. The challenge to the software exclusion of Article 52 would prove to be a poke into a 'hornets' nest'.

To many, the IBM decisions were the formal end to the software exclusion of the European Patent Convention. The fiction of a technical character, contribution or effect was severely criticised. Can technical effect indeed be a separate criterion from industrial applicability of usefulness? Can one occur without the other? Was the patent community drafting its own laws?

The European Open Source community suspected the capture of software development by multinational companies: inadvertent patent infringements would become prevalent (infringement actions for open source products naturally are easier to pursue) and restrictive licences would eventually reign. The EPO proposal to delete 'computer programs' from the list of Article 52 exclusions was rejected at the EPC revision conference in 2000. The European Commission suddenly saw a need to improve legal certainty and unity of law (something IBM had just done for them) with a Proposed Directive 'on the patentability of computer-implemented inventions' (presented 20 February 2002). The policy advanced in the

Explanatory Memorandum to the Proposed Directive remained curiously ambivalent. On the one hand, vigorous US patenting strategies are seen as indicators of a vigorous innovation culture Europe ought to compete with. On the other hand, a low threshold to obtaining patents may impose serious costs on an economy, even stifle innovation.

Keith Beresford (2000) has shown in detail that exclusions on patentability under the European Patent Convention are already almost equivalent to the liberal US practice; i.e. a European patent can be obtained on almost any 'invention' for which it would be available in the US. The 'technical effect' criterion that is supposed to distinguish European practice is not specific to examination of software but a feature of EPO examinations for inventions in every field. As Beresford writes (2000, p. 54): 'This is how the patent system has always worked: the disclosure of the invention and its reduction to practice must justify the scope of claim the applicant seeks'. Showing 'technical effect' is a requirement on claim writing, as per Articles 83 and 84 EPC and Rules 27 and 29 of the Implementing Regulations.

(1) the claims must define the matter for which protection is sought in terms of the technical features of the invention; and

(2) the description must support the claims and disclose the invention in such terms that the technical problem and its solution can be understood; in other words there must be a technical effect.

The main difference between US and European patentability requirements on software implemented business methods amounts to little more than that European applications might have to disclose the algorithm that enabled the computer to perform a complex administrative task. Arguably, the US Signature Financial Group patent (No. 5,193,056) for a business scheme for managing a portfolio of assets in partnership for economies of scale and tax efficiency (upheld in the 1998 State Street decision) might have been achieved in Europe.

The Proposed Directive attempts to confirm the status quo of (limited?) software patentability by legitimizing a vaguely worded 'technical contribution' requirement (Art. 2(b)) that is intended to prevent 'business method' patents. Can the inventive step in a computer implemented business method lie in the underlying scheme, or must it be a feature of the software? The Proposed Directive says little on this matter. It also appears to retreat from current EPO practice by prescribing the form of claims as 'programmed apparatus or processes running in such apparatus' (Art. 5), thus perhaps disallowing direct claims to 'a program for a computer'

This special issue inevitably starts from, but tries to lift its head above, the lobbying pulpit of the Proposed Directive. It aims to contribute to the question of the appropriate scope and consequences of software protection, which should precede any serious legislative effort. The remainder of this article introduces contributed papers in three sub-sections.

3. Patenting Software: Practice, Economics and Prospects

The world of innovation does not arrive in legal concepts. International conventions and national laws prefer to withhold a definition of invention. 'Anything under the sun made by man' indicated the US Supreme Court in Chakrabarty, apart from 'laws of nature, physical phenomena, and abstract ideas' (Diamond v. Chakrabarty; 447 U.S.303; 1980). Not discoveries, scientific theories and mathematical theories, aesthetic creations, not mental acts, games, doing business, programs for computers, not presentations of information, says the European Patent Convention (Art. 52). The doctrinal debate about software patents is so heated not because of the vagaries of the concept of invention but because parties fundamentally cannot agree about the purposes and appropriate scope of protection. Why are we to reward creative effort that appeared to take place anyway? In the technology boom of the 1990s, intellectual property was increasingly seen as intellectual capital driving a new economy. Multinational companies installed profit centres, offsetting the considerable costs of patenting with licensing and cross-licensing revenues. We need to understand software patents within this commercial context.

3.1 Legal and Systemic Constraints

At the start of the symposium, Keith Beresford, author of the leading study 'Patenting Software Under the European Patent Convention' (2000), exposed the concept of 'technical effect' Under EPO practice, a technical effect is present in most innovative software, for example in higher speed, more economical use of memory, more efficient database search strategies, a more effective data compression algorithm or an improved user interface. Software inventions that have been patentable include image processing, manipulation of graphics, natural language recognition and processing, a screen interface for a business management system, file distribution for network traffic, neural networks, word processors and data base management. Beresford gives examples of good disclosure and claim writing; he also reflects on infringement issues. Beresford's contribution 'Demonstrating Technical Effect in Software Cases' is available in the Comment section as a slide presentation.

The approach of the EPO to examining software related invention has been summarised as follows.

1. Identify the closest prior art.
2. Identify the difference between the subject matter claimed, considered as a whole, and the piece of prior art.
3. Identify the effect of the difference within the subject matter as a whole.
4. Deduce the problem to be solved by the intervention.
5. Analyze the problem and the solution and identify the skills necessary to understand what is realised and how.
6. Verify if the skills lie exclusively in non-technical fields - for example, mathematics, linguistics, pure programming.
7. If the conclusion is that the subject-matter claimed does not provide a solution to an objective technical problem, the objection is made:

'The claimed subject-matter is not regarded as an invention within the meaning of Article 52(1) EPC, or there is not contribution to the prior art which would involve an inventive step in the meaning of Article 56 EPC.'

An entertaining polemic against the notion of a 'technical effect' can be found in Greg Aharonian's Comment 'Why All Business Methods Achieve a Technical Effect?' Aharonian runs the San Francisco-based Internet Patent News Letter (www.patenting-art.com), and has become one of the most outspoken critics of trivial patents issued through inadequate examination. His company, bustpatents.com, offers prior art searches attacking such patents. In his Comment (which is an earlier October 2001 version of the talk he gave at the Symposium), Aharonian charges the EPO's use of 'technical' as deliberately ill-defined, lending a political shield on controversial decisions.

The regulatory requirements of patent application and grant (authoritatively presented by Beresford, controversially discussed by Aharonian) are an important part of understanding patenting practice. Equally important, but hard to penetrate are the reasons for filing and disclosing an invention. Ruth Soetendorp's article is exemplary in extracting commercial information from freely available patent databases. It also provides a step-by-step introduction to the patent system from the perspective of the insurance industry. Characteristic for the industry is a volatility of business models: computer implementations can be rapidly changed and copied. Soetendorp shows that the disclosure of innovation through a patent specification is both risky and potentially rewarding.

The first section ends with a presentation by Tim Frain, Director of Intellectual Property at Nokia. His slides provide a clear overview of the lines of conflict in the discussion surrounding the Proposed Directive.

3.2 Economic Arguments

The first section of the special issue tried to give a feel for the state of the art in patenting software. The next group of papers rehearses economic arguments in analysing the regulatory impact of patents on the software industry where innovations are incremental and cumulative; and variety of expression is less valuable than in literature, music or film.

Christian Koboldt (of London-based consultants DotEcon) gives an elegant summary of the economics of trade secrets and networks. In particular, Koboldt shows that a requirement for meaningful disclosure of software inventions (e.g. the most efficient algorithm for solving a particular problem and interface information) will discourage patent applications for desirable innovations. Software patents will be sought mainly for ideas that are trivial, and thus not suitable to trade secret protection - an adverse selection process. The main effect of making software patents available more easily would be friction in the system, aggravated by defensive patenting - a natural response to patent interference.

At the symposium, most participants agreed that the disclosure of software inventions tends to be less than transparent, even obfuscating. From an industry perspective, the incentive effect of

prospective control dominates. Whether tighter examination and disclosure requirements (the 'patent quality' argument) are a viable option here remains to be seen.

It is notable that the Proposed Directive has some concerns about the effects of software patents on interoperability. According to Article 6, the exception (under the Software Copyright Directive (91/250)) permitting decompilation of programs in order to establish interface information cannot be overridden by software patents. Does this introduce a harmonised 'experimental use exception' into European patent law?

Robert Gehring's article 'Software Development, Intellectual Property, and IT Security' shares much of the economic analysis with Koboldt: perhaps a sign that economic thinking in this area is maturing. Gehring's focus, however, is the effect of patent protection on software security. As a trained software engineer, he is acutely aware of the epistemic and practical limits of constructing and testing new programs. Code will remain unreliable and insecure. Limiting product liability is thus an ongoing concern of software developers, to the detriment of users. Gehring advocates the Open Source process of software development as the best chance for improving software quality. Open Source programs however are particularly vulnerable to patent infringement proceedings. Gehring concludes with a pragmatic proposal for a 'source code privilege' (first presented in Lutterbeck, Horns and Gehring, 2000) that would immunise every software provided as source code from patent litigation. Commercial exploitation would remain subject to the usual constraints of patent interference and licensing.

The section ends with two industry comments, one from a patent attorney who has written applications for big firms (such as IBM), the other from an association of smaller software developers.

Simon Davies of D Young & Co starts with a useful summary of European decisions and consultations in the run-up to the Proposed Directive. He characterises lobbying by the Open Source movement as 'based on the mistaken premise that deleting the computer program exclusion would open the flood-gates to software patents (rather than simply represent a clarification of existing practice)'. According to Davies:

> '[p]erhaps the most useful conclusion to come out of the consultation was that no-one in Europe was particularly supportive of the patentability of (non-technical) business methods'.

In response to Koboldt's analysis of trade secrets, Davies argues that 'it is unwise to place too much reliance on trade secret protection in an industry notorious for high staff turnover' In response to Gehring, Davies characterises the two most important factors compromising security as (i) the laxity of users in applying security updates, and (ii) the homogeneity of software platforms. Open Source cannot address the first, and is subject to the same homogenising pressures regarding the second. Davies sees patents as already woven into the fabric of the computer industry but there should be room to address Open Source concerns,

for example through Gehring's source code privilege. Industry wants both options to be available.

Sylvain Perchaud, president of the European Association of Shareware Authors, compares programs to a 'unique arrangement of bricks, which are algorithms'. Software products rely on earlier bricks, and constant redevelopment. Citing one of the few empirical studies, Perchaud argues that 75% of customers replace their software every year and almost 50% every six months (Blind et al. 2001). Smaller developers don't have the personal resources, nor prior art data bases to check, in respect of each line of code, whether an algorithm might have been used that is already covered by a relevant patent. The language of patent specifications and claims also tends to be problematic: devoid of useful information, such as the source code, while seeking broad cover that may exclude alternative technological solutions. Finally, Perchaud pleads for a sui generis software right of shorter duration, and mandatory publication of the source code.

3.3 Prospects for a Licensing Economy

The special issue ends with a rather bleak assessment of the prospects of SMEs in a licensing economy. Summing up a lifetime of research into the role of information in innovation, Stuart Macdonald of Sheffield University concludes that ' [t]hose who reap most benefits from the patent system are not those who incur most costs' Macdonald places software patents into the context of other technologies and the historical sweep of the patent system. Two empirical surveys conducted during the 1990s appear to confirm that small firms remain isolated from the external sources of information for innovation that larger firms (and their lawyers and consultants) find so important. 'Nonsensical as it may sound, the patent system is essentially anti-innovative. This is not just because it assists a very specialised sort of innovation and discourages other sorts. Much more important is that the patent system satisfies the requirements of those who need to feel that innovation is controlled and contained, part of process. Most innovation is not like this at all.'

4. Concluding Thoughts

Intellectual property is best understood not as a coherent domain, but as a group of regulations. Doctrinal consistency, for example demarking ideas from expressions, or technology from intellectual activity, may not be achievable. The main reasons are epistemological (cf. Hale, 1987). How can we refer to or know anything about entities with which we have no causal interaction? The domain of the Abstract -- of Intellectual Property -- routinely presupposed in legal analysis, may contribute little to our understanding of software related inventions. Indeed, the concept of Intellectual Property itself may be little more than a rhetorical effort to benefit from the link of private property to freedom in liberal political thought.

Yet, doctrinal consistency matters. Restricted activities, such as decompiling, copying, adapting (copyright) or manufacture, sale or use (patents) together define the scope of a property right in a given subject matter. After policy decisions have determined an appropriate 'stack' of exclusions constituting the desired protection, legal concepts need to sink into fairly stable interpretations. For instance, for patents, the scope of the monopoly is given by the

interaction between patent specification and the law of infringement. As copyright does not have claims specifying a work, the law of infringement and specific exceptions (such as fair use), govern the scope of rights provided. Software developers, distributors and users must find it possible to predict the protection available, and thus manage the risk of operating in a commercial environment. To judge from the contributions assembled in this special issue of the Bournemouth Symposium, the proposed European Directive will not settle the appropriate scope of software protection.

How do we proceed from this diagnosis? Empirical studies (e.g. Macdonald's studies cited in his contribution; Tang et al. 2001; Blind et al. 2001; Ziedonis and Hall, 2001) indicate an astonishing lack of knowledge about what inventors and firms actually do with the rights they own, why they own them, and licensing strategies. There is a suspicion that we are at the threshold of a licensing economy, i.e. an economy where many innovative activities can only be undertaken after negotiating rights. A large systematic research programme in this area might eventually reveal this 'missing center' (Kahin, 2003, p. 29) of intellectual property policy. We hope that we have at least exposed the contours of the missing centre of software patents.

Notes and References

* Professor, Centre for Intellectual Property Policy & Management, School of Finance & Law, Bournemouth University; Visiting Professorial Fellow, Queen Mary IP Institute, University of London. Many thanks to Robert Gehring, Technische Universität Berlin, for improving my technical understanding of software, for suggesting references on the history of the industry, and for compiling the US patent statistics in the Appendix. A period as Guest Fellow at the social science research centre Wissenschaftszentrum Berlin (www.wz-berlin.de) in spring 2003 allowed me to eventually complete the special issue.

1. See for example Samuelson et al. (1994, section 1.2, p. 2320): 'Programs Are Machines Whose Medium of Construction Is Text'.

2. For this diagnosis, see National Research Council Report Digital Dilemma, National Academy Press (Computer Science and Telecommunications Board), 1999: http://books.nap.edu/html/digital_dilemma

3. Cf. Lessig (1999).

4. Data provided by Bessen and Hunt (2003) for all inventions using software. Besen and Hunt do not segregate 'pure' software patents since US patent attorneys typically draft as if the claims were not for a computer program but a mechanical device including software. Aharonian's estimates for pure software patents (1976-1999, www.patenting-art.com; www.bustpatents.com) show less hits for the 1980s and early 1990s but similar numbers for the second half of the 1990s, after the U.S. Patent and Trademark Office (USPTO) relaxed its examiner guidelines in 1996. While previously a claim relating to an algorithm was only accepted if a 'physical transformation' was present, the new approach examined for a necessary 'utility'.

'5. Promoting innovation through patents', Communication from the Commission to the Council, the European Parliament and the Economic and Social Committee, February 1999. Pilch (2003) argues from searches for expressions 'related to algorithms and programming' that overall the European Patent Office 'must have granted at least 20-30,000 software patents. The European Patent Office (EPO) in Munich grants patents under the European

Patent Convention (EPC) as bundles of national patents. The EPC governs the granting of patents. The Proposed Directive 'on the patentability of computer-implemented inventions' (February 2002) however will affect national laws of EU member countries which govern EPC patents after the grant (see Art. 64 EPC). The EPO is not a body of the European Union (EU) although all states of the EU are members of the EPC. EPO: http://www3.european-patent-office.org

6. On 17 June 2003, the European Parliament's Committee for Legal Affairs and the Internal Market (JURI) approved in first reading the Proposed Directive subject to amendments. For an overview of the provisions of the Proposed Directive, see Tim Frain's contribution to this special issues. Materials relating to legislative progress are available on: http://europa.eu.int/comm/internal_market/en/indprop/comp/index.htm

7. Thus the voices heard at the symposium appear to be representative of the responses to the consultation exercise launched by the European Commission in 2000. Consultation paper 'The Patentability of computer-implemented inventions' and analysis of its 1447 responses are available on http://www.europa.eu.int/comm/internal_market/en/indprop/comp/softde.pdf and http://www.europa.eu.int/comm/internal_market/en/indprop/comp/softanalyse.pdf. About 1200 responses coordinated by the Open Source initiative Eurolinux were strongly against any software patents. Of the remaining 250 respondents, academics, engineers and start-up companies generally were critical of loosening restrictions, while lawyers, established industry players and government agencies supported the application of traditional patentability criteria to software.

8. Early decisions relating to software patents or software copyright typically denied the dual character of software as text and machine, sometimes with a bizarre allusion to computer programs operating outside the laws of physics. Swiss Patent Office Decision on a computer program calculating the reinforcement required for building blocks made of concrete: 'The idea for which applicant desires patent protection, namely the general principle of setting up a program for a computer, does not represent a creation that applies natural forces or uses them in order to achieve a technical result' (I.I.C. vol. 1, No. 1/1970; quoted in Beresford, 2002, p. 11). Japanese precedent invoking software copyright finding that 'the microcomputer program for the video game consists of diverse information and orders that embodies the programmer's original and creative concepts' (6 December 6, 1982, Tokyo District Court: quoted in Seeman, December 1982).

9. Gehring characterises software as a set of rules which can be applied by humans (more likely as source code) and/or computers. The rules describe, causally, how to get (as man or computer) from one state of affairs to another. ('Software ist eine Menge von Regeln, die von Menschen (eher Quelltext) und/oder Computern (eher binärer Code) angewandt werden können. Diese Regeln beschreiben kausal, wie man -- Mensch oder Computer -- von einem Zustand zu einem anderen Zustand kommt.' 'Was ist Software?', mimeo on file with the author).

10. Slee & Harris's Application, 1966, R.P.C. 194. According to Beresford (2000, p.4), this UK Hearing Officer decision is the first reported software patent decision worldwide.

11. See references in Chisum, 1986.

12. The 1966 commission 'To Promote the Progress of Useful Arts' was chaired by IBM vice president J. W. Birkenstock. According to Tapper (1983, p. 2, note 4): 'Patentability was opposed by mainframe manufacturers like IBM and Honeywell in amicus curiae briefs in Prater, Benson and Johnston, and supported by software house organisations like the

Association of Data Processing Service Organisations and the Association of Independent Software Houses.' During the 1960s, IBM had about 70% of the computer market (most software was specifically written for clients), and apparently feared that its hardware business would be disrupted by software patents (cf. Chisum 1986). After Louis Gerstner became CEO in 1993, IBM turned into a strategic software patentee, building and licensing large portfolios.

13. Copyright Office Circular No. 61, version 1964 (quoted in Hollaar 2002, ch. 2). Under the US Copyright Act of 1909, the US had a registration requirement as a prerequisit of receiving copyright protection. This seized with the Copyright Act of 1976. The Berne Convention, to which the US acceded to in 1989, does not allow 'formalities' -- a provision in place since the Berlin revision of 1908; cf. Kawohl and Kretschmer 2003.

14. The Hitachi IBM scandal, and the subsequent debate surrounding MITI's software law is reported in Seeman (April 1984). The experience of the IBM settlements may be one of the reasons why the big Japanese electronics firms today strongly support Open Source software.

15. Computer Edge v. Apple (Full Court of the Federal Court of Australia, 1984); Inkasso decision (Bundesgerichtshof, 1985); Pachot decision (Cour de Cassation, 1986).

16. Carl Shapiro's and Hal Varian's Information Rules (1998) is a good introduction to the economics of standards and interoperable networks. For a exposition of these arguments, see also Koboldt's and Gehring's contributions to this special issue.

17. In the UK case of Ibcos [1994, FSR 275], Jacob J explicitly rejected the Altai approach (separating unprotectable ideas from protectable expression), following a long British tradition regarding the idea/expression dichotomy as a fallacy (Laddie et al. 2001). Sufficiently detailed ideas may amount to a 'substantial part' of a program, making 'overborrowing' of a program's 'program structure' and 'design features' an infringement: a less formal criterion with (in this instance) a scope of protection similar to Altai.

18. GNU is a recursive acronym for 'GNU's Not Unix'; according to gnu.org, it is pronounced 'guh-NEW'.

19. Software patent figures taken from Bessen and Hunt (2003). In 1999, 2,658 application for business method patents were filed (an almost 100% increase to 1998. About 600 such patents were issued, of which about 400 were Internet based methods (Allison and Tiller 2001). Compare overall development of US patenting from Gehring's figures provided in the Appendix to this article.

20. IBM IP executive, personal communication to the author.

21. In the US, so-called Beauregard claims to computer program products (another IBM test case) have been possible since the 1996 USPTO Guidelines. How to conceptualise downloads is still contested (cf. Hollar, 2002, Chapter 5.V.D).

22. IBM IP executive; personal communication to the author.

23. Cf. Bakels and Hugenholtz (2002, p. 6): 'There is some confusion over whether the requirement that an invention be "susceptible for industrial application" (Art. 57 EPC) actually implies technical character. This confusion may be the result of differences in meaning of the term "industry" in various European languages. In French and Dutch the word "industrie" is used only for the manufacturing industry, which has a more narrowly defined technical character. But the English word "industry" really refers to any kind of industry even including the "government industry". Similarly, the German requirement that an invention be "gewerblich anwendbar" (commercially applicable) has a broader scope than the (technical) manufacturing

industry, but not quite as wide as the English word "industry".' See also Aharonian's Comment in this special issue.
24. See Explanatory Memorandum to the Proposed Directive (http://europa.eu.int/comm/internal_market/en/indprop/com02-92en.pdf).
25. The Rules are understood to be part of the Convention (see Art. 164(1) EPC).
26. Apparently, Signature Financial Group was advised that the subject matter was non-statutory in Europe. An EPO Board of Appeal member reportedly disagreed (personal communication).
27. Proposed Directive (http://europa.eu.int/comm/internal_market/en/indprop/com02-92en.pdf).
28. The source of the notorious phrase 'anything under the sun that is made by man' is a Congressional Committee Report for the 1952 US Patent Act that does not argue against limits to patentable subject matter: 'A person may have "invented" a machine or a manufacture, which may include anything under the sun that is made by man, but is not necessarily patentable under section 101 unless the conditions of the title are fulfilled' (quoted in Kahin, 2003, p. 11).
29. The late 1990s saw a flood of publications with the title Intellectual Capital, starting with Stewart (1997). For an analysis of the doubtful explanatory power of the concept, see Dean and Kretschmer, 2003.
30. In 1990, IBM earned $30 million from patent licensing; in 2000 royalties reached nearly $1 billion. In 2000, IBM was top of the patenting league with 2886 granted US patents, of which about 8% were software patents (Rivette and Kline, 2000; Bessen and Hunt, 2003).
31. Quoted from Roger Burt, Head of IP UK, IBM: 'Exclusion from patentability -Computer programs and business methods', Masterclass delivered at the Centre for IP Policy & Management, Bournemouth University (3 February 2003, manuscript on file with the author), referring to former EPO group director Anton Holzwarth's talk on the examiner's approach in the EPO Examining Group 2201.
32. For example, the Mozilla browser, the open source version of Netscape, consists in 35.526 data files in 5128 directories (Version Mozilla 1.4a, 2/4/2003, example given by Robert Gehring)
33. A further paper by Martin Kretschmer and Puay Tang 'Charting the Licensing Economy' was outlined at the Bournemouth Symposium and will be added to the special issue later in the year. It attempts to provide a state of the art review of empirical research on changing licensing practices.
34. For example, Bakels and Hugenholtz (2002) propose an inventory of patenting and licensing practice compiled by a European Patent Observatory.

Aharonian, G (2001) 'Deconstructing Software Copyright: 30 Years of Bad Logic', www.patenting-art.com/economic/economic.htm

Allison, JR and Tiller EM (2001) 'Statistical Analysis of Internet Business Method Patents', Final Report for the National Academies of Science -- STEP Board, December

Bakels, RB and Hugenholtz, PB (2002) 'The Patentability of Computer Programmes: Discussion of European-level legislation in the field of patents for software', Study commissioned by Committee on

Legal Affairs and the Internal Market of the European Parliament, http://www.ivir.nl/publications/other/softwarepatent.html

Beresford, K (2000) Patenting Software under the European Patent Convention, London: Sweet & Maxwell

Bessen, J and Hunt, RM (2003) 'An Empirical Look at Software Patents', Research on Innovation working paper, http://www.researchoninnovation.org/online.htm#sw

Blind, K, Edler, J, Nack, R and Strauss, J (2001) Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen: Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation, Fraunhofer Institute

Karlsruhe und Max-Planck-Institut München, http://www.bmwi.de/textonly/Homepage/download/technologie/Softwarepatentstudie.pdf

Chisum, D (1986) 'The Patentability of Algorithms', Pittsburgh Law Review 47

Cohen, JE and Lemley, MA (2001) 'Patent Scope and Innovation in the Software Industry', California Law Review 89

CONTU (1975-78) National Commission on New Technological Uses of Copyrighted Works, final report 31 July 1978, http://www.digital-law-online.com/lpdi1.0/home/CONTU/index.html

Dean, A and Kretschmer, M (2003) 'Can Ideas be Capital? Factors of production in the post-industrial economy: a review and critique', Best Papers Academy of Management, Seattle August 2003, www.cippm.org.uk/pdfs/capital_amr.pdf

Hale, B (1987) Abstract Objects, Oxford: Blackwell

Hollaar, LA (2002) Legal Protection of Digital Information, BNA Books, http://www.digital-law-online.com

Johnson, L (2002) 'Creating the Software Industry', in The Start of the Software Products Industry', IEEE Annals of the History of Computing, Vol 24

Kahin, B (2003) 'Information process patents in the U.S. and Europe: Policy avoidance and policy divergence', First Monday, volume 8, number 3 (March 2003), http://firstmonday.org/issues/issue8_3/kahin/index.html

Kawohl, A and Kretschmer M (2003) 'Abstraction and Registration: Conceptual innovations and supply effects in Prussian and British Copyright (1820-50)', Intellectual Property Quarterly 2003(2), www.cippm.org.uk/pdfs/prussia_uk7.pdf

Laddie, Prescott & Vittoria (2001) The Modern Law of Copyright (3rd ed.), London: Butterworth

Lessig, L (1999) Code and Other Laws of Cyberspace New York: Basic Books

National Research Council (1999) Report: Digital Dilemma, National Academy Press (Computer Science and Telecommunications Board) http://books.nap.edu/html/digital_dilemma

Lutterbeck, B, Horns, A and Gehring, R (2000) 'Sicherheit in der Informationstechnologie und Patentschutz für Software-Produkte - Ein Widerspruch?', opinion commissioned by the German Federal Ministry for Economics and Technology (BMWi)

Pilch, H (2003) European Software Statistics, http://swpat.ffii.org/patents/stats/index.en.html, 8 May 2003

Rivette, KG and Kline D (2000) 'Discovering New Value in Intellectual Property', Harvard Business Review (January-February)

Samuelson, P, Davis R, Kapos MD and Reichman JH (1994) 'A Manifesto Concerning the Legal Protection of Computer Programs', Columbia Law Review 94(8)

Seeman, R (1982-87) Japan Law Newsletter, www.japanlaw.com

Shapiro, C and Varian H (1998) Information Rules, Cambridge, Mass.: Harvard Business School Press

Stewart, TA (1997) Intellectual Capital, New York: Currency/Double Day

Tang, P, Adams, J and Paré, D (2001) 'Patent Protection of Computer Programmes', Study commissioned by European Commission Directorate-General Enterprise http://www.aerosme.com/download/softstudy.pdf

Tapper, C (1983) Computer Law (3rd. ed.), London and New York: Longman

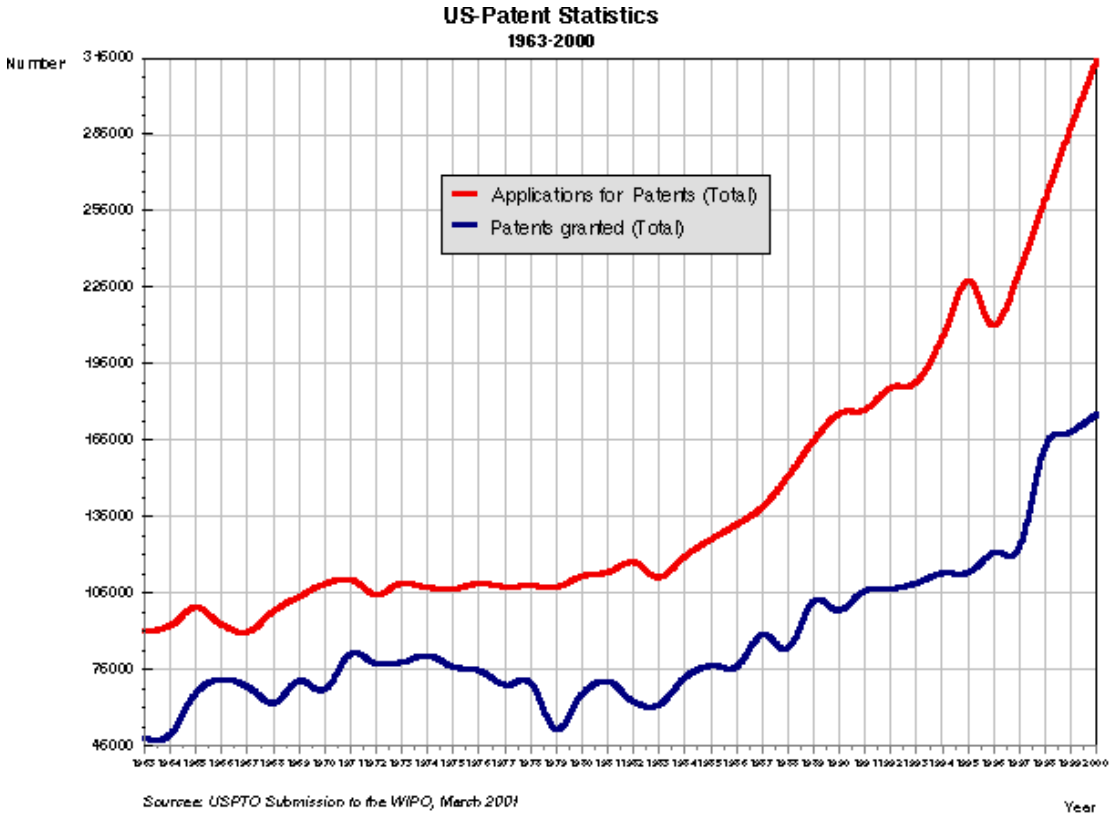
Ziedonis, RM and Hall, BH (2001) 'The Effects of Strengthening Patent Rights on Firms Engaged in Cumulative Innovation: Insights from the Semiconductor Industry,' forthcoming in Libecap, Gary (ed.), Entrepreneurship, Innovation, and Economic Growth, http://emlab.berkeley.edu/users/bhhall/papers/HallZiedonis01%20libecap.pdf

Appendix

US Patent Statistics (supplied by Robert Gehring)

Year	Applications for Utility Patents	Applications for Patents (Total)	Utility Patents granted	Patents granted (Total)
1963	85869	90982	45679	48971
1964	87592	92971	47375	50389
1965	94629	100150	62857	66647
1966	88525	93482	68405	71886
1967	85697	90544	65652	69098
1968	93471	98737	59103	62713
1969	98750	104357	67559	71230
1970	103175	109359	64429	67964
1971	104729	111095	78317	81790
1972	99298	105300	74810	78185
1973	104079	109622	74143	78622
1974	102538	108011	76278	81278
1975	101014	107456	72000	76810
1976	102344	109580	70226	75388
1977	100931	108377	65269	69778
1978	100916	108648	66102	70513
1979	100494	108209	48854	52412
1980	104329	112379	61819	66170
1981	106413	113966	65771	71063
1982	109625	117987	57888	63276
1983	103703	112040	56860	61982
1984	111284	120276	67200	72650
1985	117006	126788	71661	77245
1986	122433	132665	70860	76862
1987	127917	139455	82952	89385
1988	139825	151491	77924	84272
1989	152750	165748	95537	102533
1990	164558	176264	90364	99076
1991	164306	177830	96513	106698
1992	173075	186507	97444	107394
1993	174743	188739	98343	109747
1994	189857	206090	101676	113587
1995	212377	228238	101419	113834
1996	195187	211013	109646	121697
1997	215257	232424	111983	124068
1998	243062	260889	147521	163147
1999	270187	288811	153493	169094
2000	295926	315015	157497	175983

Last Updated on June 24, 2003


US-Patent Statistics
1963-2000
Number
345000
285000
256000
225000
195000
165000
135000
105000
75000
45000
Applications for Patents (Total)
Patents granted (Total)
Sources: USPTO Submission to the WIPO, March 2001
Year