UPGRADE
The European Online Magazine for the IT Professional
http://www.upgrade-cepis.org
Vol. II, No. 6, December 2001

UPGRADE is the European Online Magazine for the Information Technology Professional, published bimonthly at http://www.upgrade-cepis.org/

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, http://www.cepis.org/) by Novática (http://www.ati.es/novatica/) and Informatik/Informatique (http://www.svifsi.ch/revue/)

Chief Editors
François Louis Nicolet, Zurich <nicolet@acm.org>
Rafael Fernández Calvo, Madrid <rfcalvo@ati.es>

Editorial Board
Peter Morrogh, CEPIS President
Prof. Wolffried Stucky, CEPIS President-Elect
Fernando Sanjuán de la Rocha and
Rafael Fernández Calvo, ATI
Prof. Carl August Zehnder and François Louis Nicolet, SVI/FSI

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

Cover page designed by Antonio Crespo Foix, © ATI 2001

Layout: Pascale Schürmann

E-mail addresses for editorial correspondence: <nicolet@acm.org> and <rfcalvo@ati.es>

E-mail address for advertising correspondence: <novatica@ati.es>

Coming issue:
"**Knowledge Management**"

Open Source / Free Software: Towards Maturity

Guest Editors: Joe Ammann, Jesús M. González-Barahona, Pedro de las Heras Quirós

Joint issue with NOVÁTICA and INFORMATIK/INFORMATIQUE

Software Patentability with Compensatory Regulation: a Cost Evaluation

Jean Paul Smets and Hartmut Pilch

Europe is preparing major changes in its patent system. The European Patent Office (EPO) has proposed to remove limitations on patentability, such as the exclusion of computer programs in Art. 52 of the European Patent Convention (EPC). A report by the French Academy of Technologies supports this proposal but suggests additional regulation measures in order to reduce potential abuses of software patents. In this article, we try to assess the costs of such regulation measures. They add up to an estimated 1–5 billion € per year for the European Union. Various regulation approaches and cheaper legislative approaches are compared.

Keywords: Software Patents, Copyright, European Patent Office, Small Software Publishers, Cost of Regulation

1 A Drive for Patenting Logical Functionalities in Europe

Europe is preparing major changes in its patent system. Among proposed changes <http://www.european-patent-office.org/news/headlns/2000_08_17.htm>, the European Patent Office (EPO) has asked for a removal of the exclusion of computer programs in Art 52(2) EPC together with all other explicit exceptions to patentability (intellectual methods, business methods, mathematical methods etc.), in order to put the written Law in line with its recent policy of granting patents for all practical and repeatable problem solutions <(http://swpat.ffii.org/vreji/papri/jwipschar98/indexen.html>, so as to *clarify* the legal status of approximately 30,000 software patents granted under this policy <http://swpat.ffii.org/vreji/pikta/index.en.html> and to *harmonise* the practise of national jurisdictions, some of which have been reluctant to join the EPO on its quest for new frontiers of patentability.

Multinational companies in the fields of telecommunications (e.g. Alcatel, Siemens), computer electronics (ex. Thomson Multimedia), Aerospace & Defence (ex. Dassault, Matra, Thales), software (e.g. IBM) have been continuously lobbying European governments in order to get software patents fully legalised in Europe. The European Patent Office, most national patent offices as well as industrial property professionals are equally lobbying European governments[1].

Plans to legalise software patents in Europe have met opposition from various groups. 300 small software publishers from the Eurolinux Alliance have raised more than 80,000 signatures through an electronic petition[2]. Most political parties in Europe have taken public positions against software patents.[3] In some countries, parliaments have even blocked governments from taking pro-patent positions within international bodies[4].

Multiple government-sponsored studies have been conducted in Europe to assess the economic and juridical impact of software patents[5]. Reports written from an economic perspective tend to conclude that the introduction of patents in the software economy would have no impact at best but would probably lead to less innovation and competition. Reports written from a patent law perspective tend to conclude that there should be no explicit limitation of patentability and that the European practice should come in line with that of the United States of America. Some reports (IPI and the two German studies) combine both conclusions: software patents probably have a negative economic impact but should be legalised. Three reports (DG-MKT, IPR Helpdesk, Frauenhofer) include a review of opinion among software companies which shows that an overwhelming majority (80% to 90%) of software companies is happy with copyright and more or less hostile to software patents, with the noticeable exception of large software corporations (ex. IBM, Microsoft). Only one report <(http://www.proinnovation.org/rapport brevet/brevets_plan.pdf> carefully explores

Dr. Ing. Jean-Paul Smets-Solanes is currently CEO of Nexedi, a software service company based in Lille (France) which specialises in Open Source / Free Software solutions for small and medium businesses. Jean-Paul Smets-Solanes is a former civil servant of the French Ministry of Economy, Finance and Industry. He coordinated a report on software patents published in 2000 by the *Conseil Général des Mines*, a strategic body of the French Ministry of Economy, Finance and Industry (http://www.proinnovation.org) and organized in cooperation with the French Embassy in Japan the first Europe-Japan conference on Open Source and Free Software in 1999. <jp@smets.com>

Hartmut Pilch, MA in Chinese and Japanese philology and linguistics, is state-examined translator for Chinese, Japanese, English and German, fluent in several more languages, living in Munich, regularly working in EPO and other patent contexts as simultaneous interpreter. Engaged in hobby programming, especially using Lisp, since 1988, using GNU tools on SunOS since 1990, on Linux since v 0.9, co-founder of FFII.org in 1998 and Eurolinux.org in 1999, working in the area of interpreting as well as internationalisation and localisation programming.
<phm@a2e.de>

alternative scenarios to the patentability of logical functionalities: full exclusion, reduced enforcability, sui generis rights.

Among recent emerging solutions to the ongoing debate, some groups such as the French Academy of Technologies have proposed[6] to quickly and fully legalise the patenting of "computer-implementable inventions" (logical functionalities) and at the same time adopt regulation measures in order to reduce the potential abuses of such patents. The purpose of this article is to assess the financial costs of this approach and compare it with other approaches.

2 Patenting Logical Functionalities under the European Patent Convention

Most people believe that software patents relate to software. Most people also believe that software patents are meant for software developers who wish to protect their software from plagiarism. These two common ideas are quite wrong. Software patents are granted to people who do not necessarily develop and publish software. Also, software patents do not protect software authors against imitation[7].

2.1 Copyright *vs* Patents

The debate about software copyright *vs* software patents is not a debate about whether the programmer should be entitled to control the use of his intellectual achievement but about where this achievement lies – in the functionalities or in their creative combination into a complex work – and how it can be protected in such a way that the protection does not defeat itself.

For example, what the reader is currently reading is a textual composition of concepts, such as argumentation chains and rhetorical forms. Much effort went both into designing these concepts and combining them into a structured work. Copyright protects the original combination of concepts which defines this work: we hereby grant the reader a license to produce integral facsimile copies of this article but we forbid plagiarism or reusing parts of this article without permission. Copyright does not protect however the innovative or less-innovative concepts which this article is based on: readers are free to write original articles based on the same argumentation chains or rhetorical forms. The main argument for not granting monopolies on concepts or ideas in our societies is: promoting creation. Copyright would actually become quite useless if authors had to ask permission to hundreds of concept owners each time they wanted to create and publish an original work. Protecting ideas and concepts would just act as a barrier to creation.

Just like this article, a computer program is also a textual composition of concepts. Rather than argumentation or rhetorics, computer programs are based on logical functionalities. Software copyright protects the original combination of logical functionalities but not the logical functionalities *per se*. Advocates of software patents suggest[8] that designing a functionality is the important part of a computer program, the rest consisting mainly of easy "coding". Opponents to software patents on the contrary argue[9] that logical functionalities tend to be fairly easy to design, whereas the bulk of the skill and sweat of a programmer goes into weaving these logical functionalities together into a well-formed tissue[10]. In either case, subjecting software to both patents and copyright means allowing the lesser achievement to stand in the way of the greater one, thereby risking to stifle innovation rather than promote it.

2.2 Description and Claims

In order to understand what a software patent is, one should first understand how patents work. A patent contains two important parts:

Claims are the most important part of a patent since they define how the patent may be enforced in case of dispute. For example, new applications of an existing patented molecule may be claimed in a new patent by another inventor. The owner of the initial patent may produce the patented molecule but will need a license in order to be allowed to use it for his newly patented application.

One should note that patentable inventions in chemistry, mechanics etc. may contain in their description some steps which are achieved under program control. *Such inventions are not considered here to refer to software innovations*, as long as the invention-relevant problem solution involves physical causality and not merely logical functionality.

2.3 Describing Logical Ideas and Claiming Physical Objects

Patenting software innovation often leads to patents where the teaching in the description and the claimed objects are only loosely related to each other.

1. the teaching may consist of a series of steps to logically transform numbers, data or other schematic entities (algorithm, functionality, mathematical function, data format, communication protocol, language)
2. the claims may refer to "methods" (ex. image display, medical diagnosis, resource allocation, fuel saving, steel cutting, oil drilling), "systems" (ex. programmed computer, integrated circuit, telephone, missile), "computer program products" (e.g. software on disk or offered for download), or even computer programs as text structures independent of storage media.

While the description and the main claim often disclose an abstract "rule of organisation and calculation", the dependent claims may describe a wide range of concrete objects or processes. Usually the description will use terminology of microelectronics (ex. associative memory) or "software engineering" (ex. database) rather than their mathematical equivalents (ex. indexed set) or their business administration equivalents (ex. directory book). In practise this does not restrict the application of the patent claims, because "software" is the standard way of putting program and business logic to work, and the metaphorical language of "software engineering" is an equivalent of mathematical language. Where this still is not enough to cover a competitor's similar solution, patent courts may further enlarge the claim scope by applying the "doctrine of equivalence".

Thus, software patents on the description side teach us nothing but intellectual methods (logical functionalities, rules of

organisation and calculation), while on the claim side they may succeed in monopolising a broad range of material objects and processes.

2.4 Software Patents in Europe

The legal rules of patentability in Europe are quite simple. The European Patent Convention (EPC), which defines substantive patent law in Europe, contains explicit exclusions to patentability in the field of software, mathematics and other abstract innovations. The EPC does not exclude however to grant patent on inventions which make use of software, mathematics, etc. The drafters of the EPC convention had in mind the fact that technical inventions (e.g. in chemistry, mechanics, etc) which make use of a computer should be patentable as long as the innovative solution does not lie in logical functionalities (algorithms) typically thought out by a mathematician or programmer during a bathtub session, but rather in the physical causalities that form the subject of empirical laboratory research. This approach was clearly explained in the EPO's Examination Guidelines of 1978 <(http://swpat.ffii.org/vreji/papri/epo-gl78/indexen.html>, written shortly after the EPC went into force and the EPO was founded:

> *A computer program may take various forms, e.g. an algorithm, a flow-chart or a series of coded instructions which can be recorded on a tape or other machine-readable record-medium, and can be regarded as a particular case of either a mathematical method (see above) or a presentation or information (see below). If the contribution to the known art resides solely in a computer program then the subject matter is not patentable in whatever manner it may be presented in the claims. For example, a claim to a computer characterised by having the particular program stored in its memory or to a process for operating a computer under control of the program would be as objectionable as a claim to the program per se or the program when recorded on magnetic tape.*

The EPO has never had a definition of what is "technical". The question was treated as a matter of examiners' intuition. "It is hard to define what is a camel, but when you see it you know it" was one common excuse for this lack of rigor. Yet there seemed to be a reliable consensus of customary law on what is technical. German courts were not satisfied with intuition and created a definition, which has been in use for 30 years now:

> *An invention is a teaching on how to use controllable natural forces to achieve a causally overseeable success which is without mediation by human reason the immediate result of controllable natural forces.*

If applied seriously, this rule excludes any patents on what German courts call "rules of organisation and calculation" or "calculating programs for computers". However applying this rule in practise is not always easy. The EPO Guidelines of 1978 exhort the examiner to

> *disregard the form or kind of claim and concentrate on the content in order to identify the novel contribution which the alleged invention claimed makes to the known art. If this contribution does not constitute an invention, there is not patentable subject matter. This point is illustrated by the examples … of different ways of claiming a computer program.*

Thus, even if a process of cutting steel <http://swpat.ffii.org/vreji/papri/bghwalzst80/indexen.html> is claimed, there is not necessarily an invention. The examiner must disregard the claim language and look where the contribution really lies. Are the physical properties of steel part of the problem solution? Or is there only a logical problem, which is already solved at the level of logics before it is applied to steel, just like a method for adding numbers may be applied to apples or tomatoes? In the latter case there is no invention and therefore nothing to examine for "novelty" and "inventive step". This examination of seeing through the claim language and penetrating right to the core of the contribution, also called "core theory", has helped patent offices to avoid a lot of tedious and costly searching work.

Meanwhile, the EPO has however adopted a different approach[11] to the EPC. In 1985, the EPO deleted the above-cited definition of "computer program" from the Examination Guidelines and replaced it with ambiguous wordings. The intuition on what is technical also faded away. In 1998, Mark Schar, a leading EPO judge, redefined <http://swpat.ffii.org/vreji/papri/jwip-schar98/indexen.html> the "technical invention" as "any practical solution", explicitly rejecting the German definition which required a direct use of physical forces. Software patents were granted as long as no literal mention of "computer programs" was made in the claims. Starting in 1997, even claims to "computer program products" were granted, and "computer programs" were no longer considered to be "computer programs as such" if they could be said to have a "technical effect" such as making computing processes more efficient.

However there is also a tendency in the EPO to refuse at least some software patents on grounds of not having a "technical effect". This was the case with a "pension benefit system" in 2000. In such cases, the EPO first conducts a prior art search. Then, in the context of assessing "inventive step" (non-obviousness), it demands that there be a "technical solution to a technical problem", based on EPO intuitions. In a recent press release <http://www.epo.co.at/news/pressrel/2001_08_13_e.htm>, the EPO has furthermore signalled that it is overloaded with search requests for business methods and therefore will not conduct novelty searches if it is evident from the beginning that the contribution is not "technical". This again seems to indicate a certain willingness to return to the "core theory", i.e. the common sense of looking for a technical teaching before examining the novelty of that teaching, and to return to an understanding of "technical character" which no longer includes all "practical repeatable solutions" but depends on the law and on an intuitively perceived "technical character". Therefore, at present, words should be carefully chosen whenever filing software patents at the European Patent Office. For example, it is better to use the word "database" than equivalents like "indexed set" or "directory book" [12].

One should note that software patents granted by the European Patent Office have currently little value. Disputes show that high level national courts have largely enforced, even recently, the original interpretation of the EPC which excludes software

patents. Some national judges, patent examiners, academics and patent attorneys, all known for their deep involvement in the patent system, have publicly expressed[13] their worries that the current practice of the EPO could be illegal. Owners of European software patents tend not to use and enforce them because the risk of having the patent voided by a national court is much too high. This situation may sound absurd[14] but it has a great advantage since it creates some kind of self regulation which prevents abuses from software patent owners.

2.5 Business Method Patents in Europe

Typical European "software patents" are really patents on logical functionalities. Their main claims are directed to ideas such as controlling one computer by another <http://swpat.ffii.org/vreji/pikta/mupli/ep193933/index.en.html>, reading data from a medical data input source and directly displaying an analysis result to the patient <http://swpat.ffii.org/vreji/pikta/mupli/ep487110/index.en.htm> or connecting buyers to potential sellers by letting them specify an acceptable price range through an input terminal <http://swpat.ffii.org/vreji/pikta/mupli/ep762304/index.en.html>. These claims represent at the same time a functionality, a business idea, a problem solution, a programming problem and a computer program. Apart from prior art there is hardly any limit to the breadth and triviality <(http://swpat.ffii.org/stidi/frili/indexen.html> of such claims. This is because computer programs, unlike traditional patentable inventions, are not limited by the constraints of matter (physical causality) but only by the laws of the human mind (logical functionality).

The only limitation currently lies in the requirement that the business method must be automatable, i.e. "repeatable" according to Mark Schar's invention concept[15]. Thus, while the EPO granted a patent on the principle of learning languages by comparing student pronunciation with a digitally stored master sample <http://swpat.ffii.org/vreji/pikta/mupli/ep461127/index.en.html>, this covers only program logic which automates this principle, not application of the principle by human interaction between teachers and pupils.

3 Software Patents with Compensatory Regulation: System Goals

The strongest argument to legalising software patents in Europe is to put European Patent Law in line with the United States and provide European financial markets the same tools as in the United States in order to harmonise assessments methods for intangible assets in the "new economy". Many government officials in Europe still believe that there is no other way but imitate the United States and give multinational companies what they are asking for[16]. Promoting innovation, competition, small software publishers or Open Source / Free Software may still be an issue, although probably not a priority for them.

3.1 Enforce Interoperability

It is widely acknowledged that fair competition in the software economy requires interoperability, that is compatibility of file formats, network protocols and interfaces between competing products.

Software patents tend to block interoperability because they allow to create monopolies on file formats, network protocols or interfaces. For example, Apple Computer was granted an exclusive license on the Sorenson[17] patents on digital video compression. Apple markets a product called Quicktime <http://www.apple.com/quicktime> which allows to view digital video compressed according to the Sorenson method on Windows and MacOS only. Other competing products (ex. Real, Windows Media Player, KDE Multimedia, XMMS, etc.) can not view digital video compressed this way because of the exclusive licensing of the Sorenson patents to Apple.

It is therefore desirable that software patent regulation measures enforce fair licensing practices on all patents required for interoperability.

3.2 Protect Small Software Publishers

The introduction of patents in the software economy probably generates an average 30% extra cost to software development for industrial property assessment. This extra cost can be very high for a small software publisher because it requires new skills and it includes a highly unpredictable risk factors.

As a reminder, most packaged software is developed by small teams of developers, typically 1 to 5 engineers. Training existing engineers to industrial property is long and reduces their availability for innovation tasks. Hiring a full time industrial property expert is too costly. Sharing a full time industrial property expert is difficult. Fully outsourcing industrial property is often inefficient thus too expensive. Industrial property is often experienced as a useless burden rather than anything else by small software publishers as the Fraunhofer study showed.

The risk for a small software publisher is tremendous. All currently published software likely include software patent infringements without their author being aware of it. Simple universal methods, such as publishing a database through a Web server, are covered by more than 10 patents, some of which are very similar. Patent attorneys are often unable to assess the risk of litigation on those patents[18]. Although probably small, this risk relates to disputes costing millions of Euros. Small software publishers suffer much more than big corporations from this risk, not only because of their smaller financial size, but also because they do not usually own large patent portfolios which large corporations use to eliminate litigations risks with their competitors.

It is therefore desirable that software patent regulation measures allow to lower the access costs to industrial property and eliminate litigation risks for small software publishers.

3.3 Protecting Small Development Service Providers

Software regulation measures should also allow to protect small software development service companies. As we explained, simple universal techniques, such as publishing a database through a Web server, are covered by more than 10 patents. A service company which provides some kind of specific integration of a database server, a web server and scripting language may face a patent infringement dispute and make its client face it. Large development service corporations (ex. IBM, CAP Gemini) are likely to provide some kind of patent

infringement insurance to their clients, thanks to their own patent portfolio. Also, large service shops can negotiate flat rate licenses with large patent portfolios. This is not however the case of small service shops for obvious reasons.

Regulation measures should provide cheap or near-free insurance services to small and independent service shops in order to let them compete with large service shops.

3.4 Protecting Free / Open Source Software

Open Source / Free Software is a key technology to promote innovation, competition, freedom and democracy in the information society. Open Source / Free Software requires individuals to share source code and publish freely downloadable programmes on the internet.

Software patents are not a threat to Free / Open Source developed by companies such as IBM, HP, etc., which constitutes the minority of Open Source / Free Software. Software patents are however a threat to the vast majority of Free / Open Source software which is developed by individuals. A substantial amount of Free / Open Source software has already been taken off the net because of patent threats. Even in Europe, large corporations have threatened individual programmers. E.g. Thomson Multimedia threatened a Norwegian who published original implementations of patented algorithms (MP3[19]). This case is quite typical: Thomson Multimedia had probably no legal grounds to start a dispute and win because the Norwegian programmer had written nothing but a computer program as such, which cannot be a patentable invention according to European law. However, a legal dispute with Thomson Multimedia is very costly and frightening. This is why Thomson Multimedia succeeded in forcing an individual programmer to remove a program from the Internet. This is also why European Linux distributions such as SuSE and Mandrake come without MP3 encoders and with ugly font rendering mechanisms for TrueType fonts[20].

Threatening Open Source / Free Software developers with software patents is quite common in the United States. Sourceforge, a web site which hosts most Open Source / Free Software projects, receives every week dozens of letters threatening to start a dispute for patent infringement. Up to now, Sourceforge has used the first amendment act in the United States, which protects freedom of speech, to reject such claims.

However, it is not certain that there is anything such as the first amendment act in Europe to protect the publication of Free / Open Source software. Exceptions to infringement, namely private use or research, may not apply to the publication of Free/ Open Source software because Free / Open Source software often constitutes a serious commercial competitor. Companies such as Microsoft consider[21] software patents as a strategic tool to fight against Free / Open Source software.

It is therefore desirable that software patent regulation measures at least allow the unhindered publication and distribution of Open Source software and eliminate legal risks for individual developers.

3.5 Discouraging Juridical Terrorism

Most software patent trials end up with a patent cancellation because most software patents are granted for innovations which can be shown to be old, if you only search for prior art long enough. However, usually software patent disputes do not end in court. Large patent portfolio holders cross-license to each other. Small software publishers must accept whatever they are asked by larger publishers because litigations costs are too high. Moreover, specialised patent litigation companies who don't write any software make a good living by using patents from small patent holders to demand well-calculated license fees from successful software publishers.

Rather than financing innovation, software patents seem to finance a litigation economy based on the filing of trivial or existing software innovations at patent offices. Small software publishers, who are essential for competition and innovation in the software economy, are the most vulnerable targets. In order to guarantee that patents do not excessively impede innovation and competition, compensatory regulation measures are deemed necessary.

4 Software Patents with Compensatory Regulation: Measures and Costs

The French Academy of Technology has proposed four regulation measures which could possibly allow to reach the goals described above:

1. A guarantee fund
2. A database of known computing solutions
3. Improved examination
4. Incentives for SMEs

The French Academy of Technology did not provide any detail on what those measures really mean. In order to assess their cost, we shall introduce hereafter original details for those 4 measures and add a fifth one: a patent insurance, which could be considered as an incentive for SMEs in combination with a guarantee fund.

4.1 Guarantee Fund

It is possible for a state to regulate software patents abuses by introducing a guarantee fund in charge of implementing a state defined policy. For example, a state owned fund could buy 1000 software patents every year, hire a patent busting (prior art search) team and act as follows:

1. use its patent portfolio to start a dispute against any patent owner who refuses to grant fair licenses on innovations required for interoperability
2. use its patent portfolio to start a dispute against any patent owner who threatens developers of Open Source / Free Software
3. use its patent busting team against any company in the cases described above
4. attack companies which practice juridical terrorism

If patents are bought at 200,000 € from national research centres and universities, 50,000 € of which are given to researchers as an incentive, if 20 persons are hired to select patents which deserve to be bought and 20 other persons are in charge of patent busting, a Guarantee Fund costs about

200,000,000 € a year. Such a guarantee fund would also optimise the effect of patents on innovation since it allows to combine property approaches with recompense approaches[22].

One should notice that 1,000 patents a year is not that much. According to the USPTO database, a company such as IBM owns more than 30,000 patents and files more than 3,000 patents every year in the United States. Those figures are probably underestimated if one considers patents owned by IBM through a subsidiary or bought by IBM from third parties. It is very likely that IBM owns as much a 100,000 patents. Therefore, 1,000 patents a year is required to cover the basic technologies of software within a few years. 5,000 patents a year would probably be required, especially if the overall number and scope of software patents increases, which is to be expected.

4.2 Algorithm Database

A large database of algorithms can help reducing the number of litigations if it takes into account articles and software distributed around the world. Maintaining a useful database requires methodologies that go beyond mere indexing of descriptions, because the same logical method can be described in many equivalent ways. At least 100 people who analyse one patent application per person/day each, 200 people who analyse publications of specialist publications, 500 people who analyse actual computer programs and several 100 people who make sure that the database is maintained in a way that makes it reasonably well searchable. This will cost more than 100,000,000 € per year. If at least the most important publications in languages like Japanese, Chinese, Russian etc is not to be ignored, the figure may have to be doubled. It must be noted that the American Software Patents Institute (SPI.org) project, which major software companies such as IBM sponsored with large amounts of money, tried to undertake this work and is meanwhile generally considered to have failed.

4.3 Improved Examination

Improving the examination process helps reducing the number of litigations. To reach a fair level of examination, it is required to add at least 5 days to the reviewing process and to duplicate the reviewing process in order to create some kind of competition and incentives between reviewer teams to eliminate as many as possible software patent applications. This costs about 500,000,000 € a year for 50,000 software patents filed (but not necessarily granted) every year. It is likely that this number of patents will be much higher if the number and scope of software patents further explodes, as may be expected.

4.4 Incentives for Small Software Publishers

Simple incentives for small software publishers include subsidies to file worldwide software patents (10,000 € for a valid patent) and training sessions on industrial property (200,000 € for each region in Europe every year). Such incentives may eventually allow small software publishers to understand industrial property and create tight relations with patent attorneys. This is also a way to increase the number of patent attorneys in all European regions. One can expect at least 5000 patents to be subsidised every year for a yearly cost of about 120,000,000 €.

4.5 Patent Insurance

Patent insurance consists in providing full protection against patent litigations for a certain rate of a company's revenue. A patent insurance negociates flatrate licenses for all its members. Members in turn are required to check that they are not infringing on a few risky patents. Members' clients are provided a full insurance against litigation. The cost of patent insurance is unknown. Patent infringement litigation costs at least several 100,000 € and usually several millions. Some companies such as IBM require a 3% rate off the company turnover in return for their portfolio. If this can be taken as an indicator, it would

Costs Summary (in 1000 €)

group	item	unit	unit cost	min units	max units	min total	max total
guarantee fund						212,000	1,050,000
	patent acquisition	patent	200	1,000	5,000	200,000	1,000,000
	patent review	man	200	20	50	4,000	10,000
	patent busting	man	200	20	100	4,000	20,000
	patent insurance	man	200	20	100	4,000	20,000
Algorithm dabase		man	200	200	1,000	40,000	200,000
improved examination						250,000	1,000,000
	1st review	patent	2.5	50,000	200,000	125,000	500,000
	2nd review	patent	2.5	50,000	200,000	125,000	500,000
incentives						170,000	1,540,000
	collective training	session	200	100	200	20,000	40,000
	patent insurance	company	10	5,000	50,000	50,000	500,000
	patent filing	patent	20	5,000	50,000	100,000	1,000,000
						672,000	3,790,001

make patent insurance cost up to 30% of each company's revenue. Patent insurance could also cost only 1% of revenue thanks to cross licensing agreements.

In the figures above, we have set up a patent insurance together with the guarantee fund and state financed the management costs of that fund. This is a reasonable solution considering that private insurances do not want at present to insure software patent litigations. We have also included subsidies for SMEs in order to generate a software patent insurance market in the mid term.

4.6 Summary and Hidden Costs

The cost of the four regulation measures proposed by the French Academy of Technology can be estimated at about 1 billion € per year for the European Union and could grow up to 5 billion € or more within 10 years as the number and scope of software patents increases. Such an increase is to be expected, considering the fact that all human ideas can be represented in the form of algorithms and put to practical use by means of Neumann's universal computer, and that computing power is continuing to double every two years and to pervade society to a degree which has by far not been exhausted.

Such costs do not include the 30% increase in software development generated by the requirement for an industrial property strategy for software publishers, nor do they include the future cost of patent insurance for service companies. Those costs are the same for everyone and are paid by the end consumer.

5 Other Approaches to Reducing Risks of Software Patentability

Regulation approaches allow to put European Law in line with US Law which may be useful to satisfy financial markets while preserving the European identity, innovation and competition. However, it is quite expensive. Other cheaper approaches exist if one considers adopting a different Law.

5.1 Explicit Boundaries on Rights Derivable from Patents

The use of patented methods for non-commercial purposes such as private use and research has traditionally been free. Moreover medical therapy and farming enjoy certain exemptions. Two more such boundaries on patent enforceability could easily be introduced: interoperability and Open Source / Free Software.

The interoperability exception should extend to software patents the interoperability principle defined in the European software copyright Law. The interoperability exception should guarantee some kind of automatic free licensing for interoperability purposes. It could be decreed for example that a free license is automatically granted for importing / viewing data from another programme, that a free license is automatically granted for exporting data to another programme unless fully interoperable open formats exist and that a free license is automatically granted to communicate to another programme through a network protocol unless fully interoperable open formats exists to communicate to that other programme. This way, owners of patents on formats and protocols can either enforce their property by guaranteeing that all implementations include fully open and interoperable import filters or network protocols or let competitors use their property for interoperability.

The Free / Open Source software exception could explicitly stipulate that the Open Source / Free Software process is a kind of R&D or even a parallel system for the promotion of innovation and diffusion and can therefore not be considered as a patent infringement. Various degrees of exemption are possible.

Such explicit exemptions produce near equivalent effects to what the guarantee fund provides.

5.2 Limitations on What Can Be Claimed

One may consider that only claims on physical device or physical processes may be accepted and that claims on software on a media carrier should be rejected.

Thus the publication of software could no longer be viewed as a direct infringement. Even if it could still constitute a contributory infringement, this reduction in liability would significantly contribute to protecting small software publishers from abusive litigations because contributory infringement provisions are much more flexible than direct infringement provisions.

Such an explicit litigation could state for example that software distributed for free on the Internet can never be considered as a contributory patent infringement as long as all patents required to use it commercially are clearly listed. Such an approach is required in our opinion in order to make the international nature of software distribution on the Internet compatible with the national nature of patent law. On the other hand, matters could be different for software distributed on physical media or to software sold on the Internet for which some kind of national nature exists (ex. credit card number). Physical devices (ex. PABX, missile) and industrial applications of algorithms (e.g. steel cutting) could receive yet another kind of treatment.

Such explicit limitations produce near equivalent effects to what an algorithm database and better examination provide against juridical terrorism. Owners of valid software patents are encouraged by this system to reach an agreement with software publishers.

An even more friendly variant would be to restrict enforceability to embedded systems, i.e. devices which are not laid out for reprogramming by the user. This method would protect much of the traditional interests of the telecommunications and industrial software patent owners while keeping those systems free of patents where an ever-increasing number of freelancers and small companies are making significant contributions to new developments.

5.3 Sui Generis Rights

This approach has been suggested by the *Conseil Général des Mines* in September 2000 as well as by Dr. Martin Mayer <(http://www.m4m.de/internet/pmsoftpa.htm> (spokesman of the Christian Democrat/Social Union in the German parliament) and many others <http://swpat.ffii.org/stidi/basti/indexen.html>. It consists in creating a special-tailored property right (*ius sui generis*) for

algorithms (teachings of logical functionality). At the same time, it would have to be made clear that patent law is also a special-tailored right, namely for technical inventions[23] (teachings of physical causality). The relationship between copyright and the algorithm law still seems unclear. One approach is to consider copyright as applicable to all intellectual creations and at the same time extend or replace it by some specialised laws in niche areas, such as software behaviour or industrial design. Some arguments for this approach are of political nature: it provides a link between an American software patent system and a European system where patents are limited to technical inventions. Moreover it provides an outlet for accommodating legitimate interests in algorithm property, as far as they may exist, in a balanced and adequate manner.

6 What Is so Special About Software?

The software economy includes properties which do not exist in any traditional industry:

1. Software is pure information: it can be published on the Internet with zero marginal distribution costs.
2. Software developed by one man (ex. the Linux operating system, the Konqueror web browser) or a group of friends (ex. the Apache Web server, the Ogg music compression format) can compete with equivalent software developed by multinational companies such as Microsoft, Thomson Multimedia or Netscape (ex. Windows, Internet Explorer, iPlanet, MP3).
3. Highly multidimensional network effects: multiple interoperability issues are coupled in software which generates much stronger network effects than in any other field.

The first two features explain why much software innovation comes from individual developers and very small software publishers based on the Internet. It is a justification for protection measures in favour those two groups. The last feature is an argument for stronger competition protection measures.

The epistemiologic nature of software is equally striking:

1. Software is logics, a hierarchy of abstract functions. Before the computer existed, software already existed. Patent law scholars were careful not to allow the patenting of the logical aspects inherent in all apparatuses, such as "operation instructions". With the advent of the computer, these logical aspects emancipated themselves from the apparatuses and the humans operating them.
2. Software is equivalent to human reason and to human language. It consists in describing a reproducible series of steps to manipulate data (abstract information). Algorithms can describe all human reasoning. While speaking Logical Language (Loglan/Lojban), human speech becomes turing-complete. Computer programs are equivalent to mathematical proofs and are validated by means of logics rather than by physical experiments.
3. The value of software ideas lies in their abstractness <http://swpat.ffii.org/vreji/papri/ist-tamai98/indexen.html>. Ingenuous software innovations, such as the Karmarkar inner-point method <http://swpat.ffii.org/vreji/papri/konno95/indexja.html>, are extremely general, leading to patents with an unoverseeably broad scope of applications. Less abstract software ideas tend to be trivial. Most software patent claims are both trivial and broad, and this phenomenon does not significantly depend on how strictly patent offices apply the rules of "novelty" and "non-obviousness". Moreover, abstractness makes novelty search all but impossible.
4. Software is reflexive (it is its own description): it can be self generated without human intervention, it can duplicate itself just like a conscious life form, it can coordinate itself with other software just like a social form. It is an objectivation of human intelligence and lives through the communication of those who understand the language in which it is written. Many software companies have names like "Active Knowledge", "Thinking Objects" and other terms drawn from "Artificial Intelligence", which reveal that software is increasingly working as the brain within our social organism and thereby takes on functions which previously belonged to human intelligence or even to law, as Prof. Lessig's book "Code is Law" convincingly shows.
5. Software is intellectual creation and even art. There are as many ways to implement a patented functionality as there are ways to apply algorithms like chromatic modulation or twelve-tone music to the creation of symphonies. Programs are often even more complex and delicate than symphonic creations. The difficulty usually does not consist in devising appropriate algorithms (building elements) but in building a well-designed "tissue", a sustainable multi-layer hierarchy of functionalities with an infinite number of choices that require skill and imagination. Therefore copyright is at least as appropriate to software as to construction plans, scientific articles, operation manuals, pieces of music and most of the other types of intellectual creations for which it is used. Even artistic creations such as multimedia works and games include software programming. Moreover, poems written in Logical Language have a particularly high aesthetic value, and even pure programming languages like Perl are occasionally used for poetry.

These properties require to pay a lot of attention from a juridical point of view. They require to consider with great care the meaning of infringement and contributory infringement induced by the use of single software or multiple software. Moreover they make it necessary to define a clear limit to the patent system in order not to let crude control mechanisms which were designed for material objects reach out into the sphere of the human mind. Some writers like Dr. Kiesewetter-Köbinger, patent examiner at the German Patent Office, have therefore argued <(http://swpat.ffii.org/penmi/bundestag-2001/kiesew/indexde.html> that software patents constitute a radical breach of basic values of our civilisation and an attack on central tenets of our (German) constitution, ranging from Art 1 "The Dignity of the Human Being is Untouchable" and Art 2 (equality principle) to freedom of expression, freedom of contract design and the requirement that the intellectual property, which in the case of software consists in the tissue, be protected rather than expropriated.

7 Conclusion

Regulating software patent abuses has a cost: between about 1 billion and 5 billions € every year for the European Union. Only public money can finance this cost, most of which consists in improving the patent reviewing process and implementing a guarantee fund. Unless a Software Patent Regulation Authority (SPRA) is explicitly created and budgeted within the the European regulations to come on software patents, we doubt that European governments will be able to finance such costs.

Other approaches to regulation exist and provide similar effects: introducing explicit exceptions and limitations in order to enforce interoperability, to protect Open Source / Free Software and to protect small publishers against juridical terrorism. Such approaches should be seriously considered and compared to regulation approaches.

Economy however is not the only issue in the case of software patents. Law consistency issues should be addressed too. For example, any extension of the patent system to software without justifications of a positive economic impact could eventually be in violation with Article 10 of the European Convention on Human Rights. Up to now, no one has ever seriously argued let alone proven that software patents have any positive economic effects[24].

Also, the epistemologic nature of software creates unique issues. For example, the ownership of software innovations generated by "invention machines" is yet unknown[25]. Failing to design a consistent and adequate patent law may achieve the opposite of "clarification" and "harmonisation": higher juridical uncertainty than ever before.

Notes

1. see European Consultation on the Patentability of Computer-Implementable Rules of Organisation and Calculation (= Programs for Computers) <http://swpat.ffii.org/vreji/papri/eukonsult00/indexen.html>, The results of the European Commission Consultation Exercise on the Patentability of Computer Implemented Inventions <http://swpat.ffii.org/vreji/papri/eukonsult00/softanalyse/indexen.html>, Software patents: will Europe roll over for the multinationals? <http://www.theregister.co.uk/content/archive/13942.html> and Mesures prioritaires pour une accélération du mouvement de l'innovation en France <http://www.medef.fr/fr/A/Adoc/A2000/A 12-07-00 mesures-prioritaires.pdf>
2. EuroLinux Alliance Petition to Protect Software Innovation in Europe <http://petition.eurolinux.org/>
3. Conservative as well as left-wing politicians have both expressed positions against software patents. <http://petition.eurolinux.org/statements>
4. On 2001–03–14, at a parliamentary hearing a majority of coalition and opposition party MPs instructed the Dutch secretary of state of economic affairs to:
   1. define workable rules for eliminating trivial software patents
   2. only after that consider allowing software patents
   3. actively promote this view in the EU

   The test criteria were to be drafted by FENIT, the dutch IT industry organisation and VOSN, the dutch Open Source organisation. On 26th July 2001, the proposed criteria were published <http://www.vosn.nl/patenten/pers200107241.html> by FENIT and VOSN:
   1. Distinguish between technical and non technical problem solutions in terms of "planned use of controllable natural forces to directly achieve a causally overseeable effect".
   2. If programs are part of an invention: publish the source code along with the claim, also the program should work.
   3. As far as programs are concerned, the patent protects only the implementation submitted in (2).
   4. The claim should be accompanied by a proof that an experiment (again defined in terms of natural forces) was necessary to arrive at the solution and that it was actually carried out.
   5. Create a suitable incentive structure for preventing undesirable patents, avoid that rejecting is more expensive than granting etc.
5. see Stimulating competition and innovation in the information society <http://www.pro-innovation.org/rapport brevet/brevets_plan.pdf>, IPI Study: The Economic Impact of Patentability of Computer Programs <http://europa.eu.int/comm/internal_market/en/intprop/indprop/study.pdf>, Intellectual Property Initiative <http://info.sm.umist.ac.uk/esrcip/background.htm>, Patent Protection of Computer Programs <ftp://ftp.ipr-helpdesk.org/softstudy.pdf>, Security in Information Technology and Patent Protection for Software Products – Expert Opinion by Lutterbeck et al. written at the order of the German Ministry of Economics and Technology <http://swpat.ffii.org/vreji/papri/bmwiluhoge00/indexen.html> and Studie von Frauenhofer und MPI über die wirtschaftlichen Auswirkungen von Softwarepatenten <http://swpat.ffii.org/vreji/papri/bmwi-fhgmpi01/indexde.html>.
6, Avis de l'académie des technologies concernant la brevetabilité des inventions mises en oeuvre par ordinateur <http://www.internet.gouv.fr/francais/textesref/avisacatec180701.htm>.
7. An explanation of this feature of software patents can be found in Chapter 3 of the Conseil Général des Mines <http://www.pro-innovation.org/rapport brevet/brevets plan.pdf> report.
8. This is regularly suggested not only by patent lawyers but also by some representatives of universitarian informatic research, such as the president of the German Gesellschaft für Informatik e.V. <http://swpat.ffii.org/stidi/lijda/giev/indexde.html> and his colleague Prof. Endres as well as some people who have transformed university research projects into private software patent licensing businesses.
9. In software, "reinventing is commonplace", because devising an algorithm is usually easier than analysing someone else's work – patent advocates "overemphasize inventing", it is argued <http://lpf.ai.mit.edu/Patents/AgainstSP/asp-toc.html>. A former Microsoft system architect and current day IT consulting guru exhorts investors <http://joelonsoftware.com/stories/storyReader$17> to discard the widespread "building-the-better-mousetrap belief": The key to success in software business is not in getting one's hand on great ideas, but in building a team of talented programmers who will know by themselves how to "convert capital into software that works".
10. The latter view is shared and explained in detail by the German Federal Supreme Court's Betriebssystem <http://swpat.ffii.org/vreji/papri/bgh1-bs90/indexde.html> decision of 1990, which also states that operating systems are not technical inventions, because they make use of known physical devices within a framework of logics which is already predefined by these devices.
11. Patent Jurisprudence on a Slippery Slope – the price for dismantling the concept of technical invention <http://swpat.ffii.org/stidi/korcu/indexen.html)>.
12. for detailed hints on how to harness this art, see "Keith Beresford – Patenting Software under European Patent Convention – 2000, Sweet & Maxwell Ltd. ISBN 0 752 006339".
13. See documentation <http://swpat.ffii.org/stidi/korcu/indexen.html>, especially Interpretation of art 52 of the European Patent Convention in view of the question, to what extent software is patentable <http://swpat.ffii.org/stidi/epc52/exeg/indexde.html> and MV.
14. Moses, the Ten Exclusions from Patentability and "stealing with a further righteous effect" <http://swpat.ffii.org/stidi/epc52/moses/indexen.html>
15. This view has been explained in an EPO document <http://www.european-patentoffice.org/tws/appendix6.pdf> of summer 2000 and reaffirmed by the EPO's president Dr. Kober in a journal article this year. Some writers pronounce themselves even more clearly

in saying that "all business methods have a technical effect and therefore must be patentable", see Greg Aharonian: How all business methods achieve a technical effect <http://www.european-patent-office.org/epidos/conf/pat eac01/programme/16.htm>, Erwin J. Basinski: Business Method Patents in Europe – A Saussurean Explanation <http://www.mofo.com/news/general.cfm?MCatID=&concentrationID=&ID=141&Type=5> and US-Urteil CAFC 1998–07–23: Algorithmen und Geschäftsmethoden patentierbar <http://swpat.ffii.org/vreji/papri/grurnack99/indexde.html>.

16. CEC Consultation Summary Report <http://swpat.ffii.org/vreji/papri/eukonsult00/softanalyse/indexen.html>: Opponents of software patents wanted swift action on the part of the Commission but the radical nature of their proposals would require substantial negotiation if the Commission were minded to pursue a restrictive policy regarding software patents.
17. Why Hasn't Apple Released Quicktime For UNIX? <http://www.slashdot.org/askslashdot/00/03/28/1835255.shtml>.
18. Intradat AG <http://www.intradat.de> is a German publisher of a popular e-commerce system. In order to assess on which of numerous US software patents in their field they could be infringing, they spent more than 30,000 USD on expert patent research and as a result received a list of 30 relevant US patents but no information on whether Intradat really infringes and whether licenses are available.
19. BladeEnc News Section <http://bladeenc.mp3.no/skeleton/news.html>
20. Patents on Font Generation by the TrueType and OpenType Standards <http://swpat.ffii.org/vreji/pikta/xrani/ttf/index.en.html>.
21. This strategy of Microsoft became known through the Halloween Documents <http://www.opensource.org/halloween/> and has since then been publicly elaborated by Microsoft representatives at several occasions, see Pierre Bugnon, Microsoft France: Petite Digression sur le Logiciel Libre <http://www.microsoft.com/france/technet/edito/def_edito.asp>, 2001–07–24 Opensource Summit and Microsoft <http://aful.org/pipermail/patents/2001-August/002284.html> etc.
22. Shavell, van Ypersele, 1998, Rewards versus Intellectual Property Rights.
23. see the concluding paragraphs of the Dispositionsprogramm <http://swpat.ffii.org/vreji/papri/bghdispo76/indexen.html> decision of the German Federal Supreme Court, which explain this very convincingly and demands that the various specialised property systems need to be kept clearly separate.
24. Existing economic arguments show that software patents promote concentrations in the software publishing economy as well as in the software consulting services. They also show that software patents are needed for competing in economic environments such as that of the US, where software patents have become an important factor. Comfortable mechanisms for international patent filing (PCT) already exist and could be improved. European companies can use them regardless of whether the particular subject matter is patentable in Europe or not.
25. Origin of the Patents by Eric Knorr <http://www.technologyreview.com/web/knorr/knorr080301.asp>.