UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org
Vol. IV, No. 3, June 2003

UPGRADE is the European Journal for the Informatics 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/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).
UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.
UPGRADE was created in October 2000 by CEPIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

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

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

Upgrade Newslist available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Coming issue:
"Software Engineering: The State of an Art"

Legal Tools to Protect Software: Choosing the Right One

Roberto Di Cosmo

This article investigates the relative merits of the different legal tools available to protect software, in the very moment when the European Community considers changing its public policy on these issues. We offer a few clear arguments on the impact that the different legal tools have on software development and innovation, and urge the readers to form their own opinion.

Keywords: copyright, free software, intellectual property, open source software, public policy, software patents, trademark.

Roberto Di Cosmo is full professor at the University of Paris 7 in France, where he is responsible of the postgraduate studies in Computer Science, after having taught for several years at the Ecole Normale Superieure of Paris, and is currently also a research fellow at INRIA. His research interests include formal methods, mathematical logic, functional and parallel programming. He has been actively involved in the reflections on the impact of computer science in our society over the past five years. <http://www.dicosmo.org>, <roberto@dicosmo.org>

1 Introduction

If you are today in the business of writing software, you may find three kind of legal tools that have the potential to protect your work from harsh competition: trademarks, copyright and patents. It is quite remarkable that, over the past years, trademark law has been uniformly applied in the domain of software, with no voice whatsoever against it, copyright law has been used for decades with not much more discussion (there is quite a bit of reactions to the means recently proposed to protect copyright, not against copyright itself), while the application of patent law in the US and introduction of patent law in Europe, when applied to software, has faced a steep adverse reaction of many stakeholders in the software business and research arena. I would like to offer my opinion, as an European academic working in Computer Science.[1]

2 Legal Tools for Software Protection: Trademark, Copyright or Patents?

Everybody intervening in the debate today agrees that we should find a way to offer inventors a fair reward for their effort, as far as such an effort produces a result which is valuable to the community. The big debate is about which, among trademark, copyright and patents, is the appropriate tool to fairly reward inventions while still protecting the general interest and free competition (many of the stakeholder officially consider the last two equivalent.)

One often hears generic assertions like "*Internet is the future, the wealth of the future is Intellectual Property, and protecting Intellectual Property is necessary; hence we need software patents*" which are quite fallacious, as there are more ways to protect intellectual properties than through patents.[2] Indeed, one can build a sizeable market share simply through trademarks, as the RedHat Linux distribution does[3], and one can build a decades long monopoly in operating systems just through copyright, as Microsoft[4] did. So, one may legitimately wonder what is really behind this sudden interest in software patents, in a climate of hurried urgency, after decades of successful software innovation under the intellectual property protection guaranteed by trademark or copyright.

3 Copyright versus Patents

First of all, one needs to understand clearly that the basics of patent law differ deeply from copyright law (for the sake of clarity, I will drop trademarks in what follows). Also, I will not enter into the technical details of the difference between US copyright law and French or European copyright law, as these are minor with respect to the differences with patent law.

3.1 What is Covered, and how much Power Is Given to the Author

Copyright covers a *specific*, *existing* program code, in the same way it covers a specific, existing novel, like, say, Arthur Conan Doyle's "The Hound of the Baskerville", from being

1. I also urge you to read Professor Ullman's remarkable opinion on the subject [4].

2. I intentionally omit arguments like "*patents are needed to get venture capital*", or the ludicrous, but authentic "*a big company from overseas sent an expert to explain to me why I need software patents to be competitive with them*".

3. For those who do not yet know it, you can perfectly legally download a full copy of a RedHat CD on the Net, at no cost. Its license agreement not only allows you to do it, but also encourages you to do it.

4. Microsoft is a very late entrant in the software patent game.

copied or misrepresented without authorization. For example, you cannot take Conan Doyle's work, change a few lines, replace his name on the front page with yours, and go selling the book for yourself.

But that's almost as much as copyright can do: Conan Doyle has not a right to prevent others from writing stories about some kind of very smart detective investigating intricate and mysterious cases, and luckily so, as otherwise, Poirot and so many other characters which are cherished by many of us whould never have seen the light (let alone the fact that, were copyright that powerful, Edgar Allan Poe could have prevented Conan Doyle from writing about Sherlock Holmes in the first place).

In the programming arena, this means that you cannot steal somebody else's program (which some big proprietary software vendors nevertheless do from time to time), *but you are very much welcome to write a program with similar functionality*. Let us say it again: you cannot take the Office code from Microsoft and make it part of a product of yours, but you are quite free under copyright law to write an equivalent, or better Office suite, like the OpenOffice project has successfully done.

This sounds quite fair at first sight: one protects the authors from being outright ripped of their original work, but at the same time, one allows free competition on the merits, by making authors to compete for the best fiction novel or the best office suite, and the cross-fertilization of ideas that is thus possible ends up increasing the global wealth, with general satisfaction.

On the other hand, *patent law* was introduced on a very different basis from copyright law: the idea was to motivate "inventors" to disclose the details of their inventions to the public, hence making them available early to foster further innovation, instead of keeping such details secret to protect their business. In exchange for a full disclosure, inventors got a precious gift: a full stateawarded

and enforced monopoly on the financial exploitation of every possible realization of their invention for a limited amount of time (effectively setting them free from the annoyance of free market competition for such period). For this to be a fair deal, everybody must comply with his part of the contract: the inventor must disclose a valuable new and non obvious invention and prove that it really works, and is not another perpetual motion chimera, or a rediscovery of warm water, in order to obtain his "patent". This is why, unlike copyright, that protects the authors of even the most ugly and useless piece of literature or bit of code, patents are not "automatic", but are awarded by a patent office which is supposed to deeply scrutinize the patent proposal before accepting it.

Is patent law well adapted to software? According to Article 52 paragraphs (2)(c) and (3) of the European Patent Convention [1] [2], the answer is a clear no. But a few powerful forces are doing their best to change this state of affairs, claiming that "no" is not the right answer, and pushing the EC to vote on a proposal of directive that would introduce software patents in Europe [3].

Luckily, we do not have to resort to philosophical thought experiments to understand what such a change in the EC directive would mean for the software business: one overseas country has long since engaged into the software patent way, so we can look at what happened there. What we find looking at the software patents that have been awarded, is that they "protect" an abstract method to solve a problem, not a specific piece of program code, that may not exist yet, written to solve the problem. Despite the overwhelming amount of sophisticated and

diverse arguments that you can find around on the subject, there is one piece of solid fact that makes such patents deeply different from copyrights for a programmer: if patent law is extended to software, then *you cannot write a program* with a functionality similar to what is covered by a competitor's patent, no matter how superior is the program you write, how badly written is your competitor's program, or, worse, whether your competitor has actually written, or intends to write such a program at all! To write your program, you need to get a licence from the patent holder, and this means that you may need to spend some money to get the patent. But it is not all about money: the patent holder may deny licensing a key technology to people he dislikes.

I believe an example is quite appropriate to make it clear what a patent can do: quite recently, a large software company has made available on the Net part of the technical specifications of a network file system, under the pressure of a judicial order, but managed to smuggle in the fine print a few sentences that prevent open source developers from writing code compatible with that specification.[5]

Would we really have accepted the notion that M. Conan Doyle was legally entitled to declare that no woman was authorised to write novels "compatible" with his, i.e. dealing with detectives solving mysteries by their deductive skills, hence effectively preventing Ms. Agatha Christie's work from ever being written?

3.2 Licence to Kill

Writing software means, in the day to day life, solving dozens of different "novel", yet similar, problems by combining a set of basic techniques that form an ever evolving "state of the art". Sorting values, arranging data into efficient memory structures, laying out clean user interfaces, balancing load on a set of processors, exchanging data between remote computers, reading from and writing to different data formats and so many other diverse tasks are part of each programmer's daily work,

5. See <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnkerb/html/Finalcifs_LicenseAgrmnt_032802.asp>

and what makes their work unique is quite often not the particular choice of the algorithm or the data structure, but the unique way this programmer turns this algorithm and data structure into actual code.

This is comforted by our daily experience in academia: when we give a homework to the students, it is often the case that, given a clearly stated specific problem, there are only a few, if not just one algorithm or data structure that provide the most efficient solution. Yet, no two programs that are turned in are identical (unless somebody violated copyright law, of course).

Now, any sophisticated enough running software system may need to use code implementing thousands of methods, ranging from naive ideas to very complex algorithms. While one may agree that sophisticated algorithms really deserve recognition, for they represent significant advances in science, it is very rarely the case that such advances are disclosed in exchange for a patent (the RSA public key protocol being a remarkably rare notable exception): most of the time such advances are published in scientific journals, for all to read, which is quite reasonable, since most of the time such results are obtained by means of public funding.

On the other hand, a worryingly large amount of the software patents that have been awarded overseas cover desperately naive ideas, like the “traveller in a hurry” heuristic to find a quick meal (patent #5,249,290), or variants of working drafts of some standardization committee (the well known Cascading Style Sheets patent awarded to Microsoft), or even trivial methods for business practice, like the *one-click buy patent* held by Amazon.[6]

3.3 Software Patents are *Special*

The big problem is, there is often no way to circumvent naive ideas, and there is no way whatsoever to ensure interoperability with patented protocols, formats or standards, if the patent owner refuses to licence them.

This means that a broad enough/vague enough/trivial enough patent is a true licence to kill competitors: consider the case of a large company facing a young dynamic start-up that came up with a truly wondrous idea, having the potential to turn them into a real competitor over a few years. The start-up may of course patent its idea. But it will never be able to turn it into a product: as soon as it tries to release the code, it will be immediately evident that such code implements a full range of trivial accessory functionality, at least a few of which are likely to be covered by some trivial patent held by the large company.

Hence, the only viable solution will be to let the large company buy-out the start-up, which may be sometimes satisfactory for the venture capital that can realize a moderate benefit, but never satisfactory in terms of competition and free market, and even less so in terms of innovation, which is dwarfed by such practices.

This is probably why one only sees a limited number of cases in court: either the contendent are both large companies, and then they just pass some agreement to cross licence their patent portfolios for free, or one of the two is small fish, and is generally eaten without too much fuss.

4 Act Now

To sum up, one big trouble with software patents, is that in the software business coming up with a patentable idea is too often trivial, while developing software that uses a truly notrivial idea may be extremely costly and complex, and usually involves too many trivial ideas that too often turn out to have been patented by some smart guy, who holds then a finger on the trigger of a powerful weapon pointed at your very business. And in such a situation, this smart guy need not even be a software company.[7]

Next time you consider whether copyright or patent law are more appropriate to software, remember that copyright law allows to protect costly developments, and it has done so for a long time, while patent law may be abused to nullify such costly developments by means of alleged violations of some obscure triviality that amounts to the software equivalent of hot water.

Then, ask yourself if it is acceptable to give a licence to kill to a bunch of large companies, and then let them free in the middle of the vast community of software developers working in the SME/SMI that are Europe’s best asset for building the society and the marketplace of tomorrow.

In case you consider that there might be a problem, you should take a few minutes of time to make your representatives in the European Parliament aware of your opinion, while you are still in time.

Acknowledgements

The author wish to thank Juliusz Chroboczeck for his interesting comments on a first version of this article

References

[1]
European Patent Convention, article 52, 1978.
<http: //www3.european-patent-office.org/legal/epc/e/ar52.html>.

[2]
Council Directive on the Legal Protection of Computer Programs (91/250/eec), 14 May 1991.
<http://europa.eu.int/eur-lex/en/ index.html>.

[3]
Proposal for a Directive of the European Parliament and of the Council on the Patentability of Computer-implemented Inventions. Official Journal C 151 E, p. 129–131, 25 June 2002.

[4]
J. Ullman. Ordinary skills in the art. Knuth Prize Award Lecture, 2000.
<http://www-db.stanford.edu/~ullman/pub/focs00.html>.

6. Let me be clear on this: Amazon did a wonderful job at online book selling, and built a truly wondrous system to back up its growing business, including the quite sophisticated implementation of a secure system enabling one-click buy for a multitude of recurrent customers. But while the implementation is sophisticated and must be protected, as it is protected by copyright law, the idea of one-click buy is a triviality.

7. The US have recently seen the birth of a peculiar kind of non-software company whose very business is just to accumulate software patents and then go after moderately successful software companies and force them to ‘pay protection’ by threatening to sue.