Review of Economic Research on Copyright Issues, 2007, vol. 4(1), pp. 3-14

PATENT AND/OR COPYRIGHT FOR SOFTWARE: WHAT HAS BEEN DONE SO FAR?

RICHARD WATT

ABSTRACT. The particular case of software seems to have stretched the patent-copyright divide to the point of breakage. Inspite of being traditionally excluded from patent, software is an obvious case of a single creation that embodies both "expression" and "innovation", and so strong arguments exist for software to be both copyrightable and patentable material. The legal profession has looked carefully at the patentability of software over the past 15 years or so, both from a fully legal perspective, and using economic-type arguments. But we are still waiting for the economics profession per sé to set to work on this issue. Here, I shall go through some of the most well known arguments surrounding the protection of software, and then put forward a personal opinion as to what theoretical economists are likely to add, if and when they include this important question on their research agendas.

1. INTRODUCTION

> "It is crystal clear that software is entitled to both copyright and patent protection. It should also be crystal clear that these forms of protection should not be mutually exclusive. There is no justification whatsoever in the Constitution, the federal statutes, or the case law to justify a denial of joint patent and copyright protection for software." (Einhorn, 1990 p. 278)

The above quote is now more than 15 years old, and in spite of that, many parts of the world are still debating whether or not software protection should or should not be extended from copyright to include patent as well. Together with this quandrum, software is now also increasingly released under open source licenses, a mechanism that seems to fly in the face of both traditional forms of intellectual property protection.

Ever since it was first conceived formally and embodied into legal doctrine, intellectual property has been divided into two principal branches – patents and copyrights. Copyrights protect expression of ideas, and patents protect innovative ideas themselves. While initially these two categories were quite likely sufficient to provide an all encompassing and mutually exclusive description of all possible creations of the mind, it is becoming increasingly clear that this is no longer true. It has been repeatedly noted by legal scholars that there exist many cases of intellectual creations that require new *sui generis* regimes (see, for example, Reichman 1994), and it is also clear that there are creations that can satisfy the basic definitions of both copyright and patent. Most notably, the case of computer software has been repeatedly discussed; see for example Samuleson et al. 1994), since software seems to provide the most important example of a creation with clear aspects of

both copyrightable and patentable material, and will surely continue to be at the forefront of the intersection between patent and copyright.[1]

The current generation of creators and consumers of creations have come to inherit a (largely) bimodal system of intellectual property rights. More to the point, historically the creators of software have been forced into the particular box of copyright.[2] Digitally speaking, software is indistinguishable from any other digital product (text, music, photos, databases, sounds, graphics, videos, etc.), and as such it is (somewhat) logical to protect software under the same regime, i.e. copyright. But, software is different as it also has a clear aspect of functionality and innovation, which has led to the argument that patent is also relevant. But in spite of the obvious innovative nature of software products, patent protection has been hard to come by, and in Europe at least, largely impossible.

The principal defining aspect of the problem is that it is not always easy to separate ideas from expression. When innovation and expression are too intermingled to be easily separated and protected independently, the protection of expression may, inadvertently, provide protection for the idea expressed.[3] From a logical standpoint, if a single creation does contain both protectable ideas and protectable expression, we would expect that each regime of intellectual property law protects different, mutually exclusive, aspects of the creation. The use of one or the other regime is fine for any creation in which the expression of ideas and the ideas themselves can be fully and completely separated from each other.[4] But, what shall we do when the expression of an idea is an integral part of the idea itself (i.e. expression and idea are in some way inseparable)? Such may be the case of software. Copyright alone either leaves important aspects unprotected, or it overprotects (due to ease of grant, and long duration, etc.). Patent alone, it is argued, leaves object code unprotected and thus, presumably, susceptible to copying, and the inventive step requirement will almost surely not be met for software developments. Using both may clearly overprotect, as some aspects will be protected twice. Additionally, we must deal with the special market-type characteristics of software, namely the huge network economies of scale, and the importance of interfaces between programmes.

[1]The fact that what is copyrightable and what is patentable can sometimes be debateable is is not really a recent development. As early as 1930 Justice Learned Hand stated "Nobody has ever been able to fix that boundary [between 'ideas' and 'expression'] and nobody ever can." It is, however, something that has become more and more accentuated with the general passage of time, with the introduction of new technologies, and with the advent of new embodiments of intellectual products (above all, the digital environment for copyrightable creations).

[2]It is interesting to ask why software was copyright protected in the first place. The reason seems to be that software was first shown to lawyers and Judges as source code, which looks just like any other literary work. Also, copyright avoided the obvious complications of deciding on inventive step, novelty, industrial application, etc. Furthermore, the precedent had already been set by other new technological forms of representation that had also been collected together under copyright. The (then) exclusionary mindset (either copyright or patent, but not both), led to rejection of patentability (see Widdison 2000 for more on the reasons why copyright was set as the default protection mechanism for software).

[3]On top of this, of course the "merger doctrine" limits the ability to protect in such cases of intermingled expression and ideas.

[4]An interesting example could be the case of a novel, where the expression of the story is simple to separate from the actual story being told. The expression is obviously copyright protected, but I do not know of any case in which the storyline itself is patented as an idea, most likely because it would not stand up to the industrial applicability requirement.

The current paper attempts to briefly summarise the way the legal academic profession has dealt with the problem of the legal protection of software, and to put forward some opinions as to how the academic economics profession is likely to go about analysing the same problem. I have deliberately set aside the issue of open source, which is delt with in the papers by Leveque and Meniere, and that of Koski in this issue.

2. The Special Case of Software: What Has Been Done?

The problem of software has been considered by the legal profession in far too many papers to mention here. However, I will mention a few of the more salient ones of the bunch, and I will also attempt to classify the types of argument that occur. First and foremost, though, I feel that it is worthwhile pointing out that in spite of the inclusion of such terms as "economics perspective", or "economic approach" in the titles of several of these papers, the analysis is not at all rigorously presented, as strict economics-types would expect.

Basically, there seem to be four approaches to the problem of the legal protection of software (and indeed of almost any type of creation that is an outlier to the current patent/copyright regime):

(1) we can argue that, under proper analysis, software can indeed be appropriately protected by copyright alone (the "*status quo*" approach);
(2) we can argue that copyright alone is insufficient, but that the existing regime of patent can appropriately complete the shortfalls of copyright for protecting software, so the two existing regimes are sufficient and workable (the "*make it fit*" approach);
(3) we can argue that for software a special *sui generis* regime is required, with some aspects of both copyright and of patent (the "*sui generis*" approach);
(4) we can argue that we should abandon the entire existing regime, and start anew with a clean slate, and find a system of intellectual property rights that can properly cater for all intellectual products, software included (the "clean slate" approach).

2.1. The "*status quo*" approach.

The "*status quo*" approach to software attempts to show that, under careful scrutiny, software can be appropriately protected by copyright alone. Ginsburg (1994) is a good example of the this type of argument. The question of whether or not copyright alone is suitable for software is usually posed as whether structure, sequence and organisation are protected as "non-literal elements" of the protected program code. A close analogy could be the copyright in a novel or play, which is quite generally recognised to extend beyond the pure language to more or less detailed elements of the plot itself. But for most such analogies, copyright is rather thinner than what is required for software.

Never-the-less, Ginsburg (1994, p. 2560) argues "... all Circuits that have addressed the question agree that, in principle, copyright protects not only literal code, but also non-literal elements of a program, such as its structure, sequence and organization." Furthermore, again according to Ginsburg, copyright can work for software since it has extended to many other items that are far more functional than pure text; "... copyright also protects architectural plans, even though these set forth instructions to construct a 'functional' object, a building. Moreover, since

the 1990 Architectural Works Copyright Protection Act, it is clear that architectural plans are infringed not only by copying in the initial format, but also by following the plans to execute the building ..." (Ginsburg 1994, p. 2567).

The main argument for continuing with the current copyright regime alone, and disallowing patent altogether, seems to be based on software innovations not being able to satisfy the incremental step requirement of current patent law anyway, and on the argument that if software were patented, then there would ensue a "patent thicket" type of problem which would inefficiently hinder further developments. Also, so the argument goes, the discretion and good judgement of the courts will ensure that copyright is used properly, extending to more than pure literal code, but not choking off the sequential innovation that clearly defines the development of software. Finally, the extension of patent to include software will almost surely lead to bad decisions on patent granting, and this would be very costly to undo. Thus, the pundants of the *status quo* regime, that is, sticking with copyright alone, can be seen to be quite the risk averse types, who predicate absolute caution of moving into the unknown.

2.2. **The "make-it-fit" approach.** This approach to the problem attempts to separate software into functional and non-functional aspects, and protect each independently (and separately) using both the copyright and the patent regimes in their existing formats. The argument is that copyright works, but it needs to be augmented by patent, and above all, the existing IP regime seems to be suitable without need for any reforms. The writings of Prof. Dennis Karjala (see, for example, Karjala 1998) provide good examples of the make-it-fit approach.

Karjala (1998) considers that once we identify those aspects of software that are purely functional, and those that are not, each can be protected by only one intellectual property regime – functional aspects by patent, and non-functional aspects by copyright.[5] Karjala argues that copyright is suitable to protect literal code (and mechanical or electronic translations thereof), but that higher level aspects, like structure, sequence, organisation and design should be considered patent subject matter. His underlying argument is that to deny patent protection means that software would be protected entirely by copyright, which would over-protect the functional aspects. Thus copyright and patent are seen to be complementary to each other, at least for the case of software.

Never-the-less, the make-it-fit approach does seem to carefully avoid the problem of unravelling expression from ideas expressed. In particular, Karjala (1988, p. 62) states "... the difficulty of drawing the line (between expression and innovation) in specific copyright cases does not relieve us of the obligation to do so.", something that economists would likely dispute! The make-it-fit approach would have us leave the problem of defining inventive step, non-obviousness, etc. to properly informed patent officials, and thereby proponents of this approach appear to skillfully avoid the problem altogether.

[5]This reflects the decision in Whelan Assoc. v. Jaslow Dental Lab, Inc. (797 F. 2d 1222, 230 USPQ 481 (3d Cir. 1986)), where the court attempted to fix the boundary between expression and idea; "The line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."

Other supporters of the make-it-fit approach are not hard to find. In particular, Einhorn (1990) is an early example of this approach (see the quote at the start of this paper). Einhorn's argument is that joint protection works since patent and copyright are "very different types of protection, each protecting computer programming at different levels of generality and to differing extents."

2.3. **The "*sui generis*" approach.** Advocates of the *sui generis* solution to software are certainly the more plentiful of the legal-type papers. It stems from an existing branch of legal literature that analyses the apparent breakdown of the existing IP regimes to deal with certain creations. For example Reichman (1994, pp. 2500-01) notes that "... the nineteenth-century vision that subdivided world intellectual property law into discrete and mutually exclusive compartments for industrial and artistic property has irretrievably broken down. The theory that the classical patent and copyright models coherently address the way intellectual creations behave has been discredited by its inability to deal adequately with the behavior of many commercially valuable, cutting-edge intellectual creations. ... Instead of a clear line of demarcation, there is a permeable line of demarcation that allows an intermediate zone of marginal subject matters to coexist within a space that the classical system formally ignores. ".

In a more recent paper, Widdison (2000) envisages that the TRIPS agreement implies that software cannot avoid some sort of involvement with patent, although the author is wary of the effects on the balance of incentives, and of the presumable increase in black market activity, if patent is applied in its current form. Widdison's argument is that a completely new arrangement is required, characterised by a protection mechanism with:

(1) no account of novelty, inventive step or industrial application, but only a demonstration of originality (just like copyright),
(2) proper protection of all aspects of a programme – internal design mechanisms, literal code, and functional characteristics,
(3) short duration, perhaps 5 years renewable annually,
(4) clear allowance for reverse engineering for the purpose of studying and understanding the concepts and techniques used, and
(5) a specific allowance for royalty license payments for all or part of the programme.

More recently, but in an unpublished paper, Barwolff (2002) argues for a *sui generis* regime that is much more closely related to copyright than to patent. Indeed he is very sceptical of the benefits of using patent, and indeed he thinks that disallowing patent should be seriously considered, or at least making the grant of patent very difficult to achieve. His suggestion is characterised by;

(1) full disclosure of interface information,
(2) copyright only if software is distributed along with source code for easier understanding,
(3) availability of patent but only with a high threshold and with knowledgeable patent examiners,
(4) compulsory licensing (at reasonable, non-discriminatory rates), and
(5) significantly limited duration.

The real fireworks on the patentability of software, however, appear in a series of papers published in a symposium issue of the *Columbia Law Review* in 1994.

The leading article in that symposium is Samuelson et al. (1994), who argue for a *sui generis* regime, although they are not quite so specific about the general characteristics of the new regime. The paper does, however, provide some very useful insights of the underlying problem, and of why neither copyright nor patent as currently defined, are suitable alone.

The key issue for software is that "... the primary source of value in a program is its behavior, not its text." (Samuleson et al., 1994, p. 2315). However copyright can only protect the textual expression, and so the implication is that copyright alone is insufficient for computer software. Indeed, Samuelson et al. go on to imply that patent protection might be more adequate, since "... programs are, in fact, machines (entities that bring about useful results, i.e. behavior) that have been constructed in the medium of text (source and object code)." (Samuelson et al., 1994, p. 2316).

Not withstanding, Samuelson et al. argue that patent protection of methods alone is also inadequate for the case of computer software, since "... patent law requires an inventive advance over the prior art before it grants protection. Protecting incremental innovations in program behavior through patent law would thwart the economic goals of the patent system: to grant exclusive rights only when an innovator has made a substantial contribution to the art and advanced competition to a new level." (Samuelson et al., 1994, p. 2346). So, the incremental nature of software development can be seen to impede the utility of this legal paradigm as a suitable protection method as well.

Hence, the argument in Samuelson et al. can be summarised as "... copyright protects only text and text is largely independent of behavior; and incremental innovation cannot meet patent standards. Attempts to stretch the bounds of existing regimes to protect the incremental innovation in software will result in either too much or too little legal protection." (Samuelson et al., 1994, p. 2365).

Therefore, according to Samuelson et al., in spite of the fact that computer software does indeed seem to qualify for both copyright and patent protection (at least in the US), neither paradigm is really suitable, and indeed "... the application of both copyright and patent law to software innovations may impair the effectiveness of both forms of protection. It has also created considerable uncertainty about the scope of protection available from each." (Samuelson et al., 1994, pp. 2346-7).

2.4. **The "clean slate" approach.** Finally, there is at least one very novel approach to the type of problem posed by software, that is not really a *sui generis* regime. A *sui generis* suggestion would be designed to fit along aside both copyright and patent, but Dreyfuss (1992) suggests that a complete overhaul of the entire intellectual property rights system itself is justified. Dreyfuss would remove both copyright and patent in their current forms, and introduce an entirely new regime that is able to cater for all types of intellectual creation. Rather than the current situation in which we have two major protection regimes, and a whole host of minor ones for specific outlying cases, we should attempt to define a new legal system, with a single regime that can accommodate each possible case – that is, attempt to have each and every separate and individual case defined as a simple "special case" accommodated within a very general protection regime.

The rather radical suggestion of Dreyfuss appears to be based on the overall inadequacy of the current regimes to deal with intellectual creations, as was argued,

for example, by Reichman (1994). It is in response to this inadequacy that Dreyfuss suggests a major overhaul; "The law has long had difficulties accommodating information products within the copyright-patent regime... modern developments make this regime unsuitable for the future... The time has therefore come to rethink copyright and patent, to unify the theory of intellectual property in order to create a system of law that is capable of providing the right incentives across the entire domain of human intellectual endeavour." (Dreyfuss 1993, pp. 233-4).

While Dreyfuss' suggestion might appear, at first sight provocative and radical, we should recall that with the onslaught of technology, and the general development of social and cultural institutions, the details of the legal system must also vary. The legal system needs to be mailable to social, economic and technological change as it occurs. What was once a seemingly logical legal detail in the definition, characterisation and protection of intellectual property in particular can very quickly become obsolete. In that vein, one can formulate objections to the legal difference that is established between patents and copyrights on several standpoints, that not so long ago would not have been so logical. Many authors have noted that the current division between patents and copyrights leaves some creations with insufficient protection, while other creations are doubly protected. This is clear evidence of an undermining of the very reason why two different laws exist in the first place.

3. What Have Economists Said?

The "economics of copyright" has dealt with the specific case of software in a great many theoretical papers (see, for example, the survey by Amy Marshall in this issue). However, while the issue of the extent of protection has been addressed, the way this protection should be structured seems not to have been studied. As a general rule, the economic theory of optimal protection introduces legal protection somewhat indirectly, and in a variety of different forms; a cost function for copiers, a stochastic indemnity (remedy) for rights holders, a demand function (more concretely, a difference between realised and potential demand), etc. Most models that are applicable to software could just as easily be referred to any other digital product, and I have not found any published papers that deal specifically, and formally, with the copyright vs. patent debate. Thus economists appear to have remained largely silent on this question.[6]

A possible exception to the lack of economists' input on the issue of the appropriate structure of IP protection for the particular case of software is a working paper by Baseman, Warren-Boulton and Woroch (1995). In that paper, the authors conclude that copyright in its current guise requires some alterations (basically to specifically exclude certain aspects of software, e.g. aspects that achieve the status of *de facto* standard, and reverse engineering should be allowed). However the paper does not mention the possibility of software patents at all.

4. What Would Economists Say?

At the risk of unjustly putting words into economists' mouths, I will now attempt to outline the type of reasoning that we could expect from a more or less

[6]One very well known economist, Paul Klemperer, has written on the topic, but rather than in a formal paper, in the trade press (see Klemperer, 2004). In that article, Klemperer advocates caution in extending software protection to include patent, and if of the opinion that software patenting in USA has "gone too far".

detailed examination of the copyrightability cum patentatibility of software under the economic theory microscope.

I think that it is fair to say that, with few exceptions, most authors who have addressed the issue of the appropriate structure of protection of software have come to the conclusion that neither copyright nor patent in exactly their current formats, are suitable. The general tactic appears to be to take the existing regime of copyright, to see how it would need to be altered in order to provide for the appropriate protection, and then see how the final result compares to the current patent format. In that sense, almost all commentators have advocated some sort of *sui generis* regime, and the only point of debate seems to be how closely related that regime should be to copyright, and how closely related it should be to patent.

A *sui generis* regime that is based on copyright and patent, would most likely be seen by theoretical economists as some sort of "convex combination", or weighted average, of two extremes. If we imagine a multi-dimensional space, defined by all of the variables involved in a full description of IP protection (e.g. duration, innovative step, applicability, novelty, etc.), then copyright and patent are just two particular points in that space. It appears to me that the clean-slate approach of Dreyfuss is really just the recognition that such a parameter space exists, and that space is what defines the limits to any protection regime. Different creations would justifiably be given different protection schemes defined by different points in the general space. If we do stick with copyright and patent as our two initial defining points, and if we are interested in a *sui generis* regime based on these two starting points, then we might imagine the straight line joining them acting as some sort of constraint to a maximisation problem, where the objective function could be some relevant definition of social welfare. Then the "optimal" protection regime for software would occur where the social indifference curves reach a tangent on the restraining line.

The problem, of course, is that social welfare is an impossible measure to ever get a very realistic handle on, and so the final solution is totally dependent upon the researcher's initial point of view. Thus, perhaps the best we can ever hope for is an informed opinion. Never-the-less, there is another way of approaching the problem, that might very well appeal to economists. As opposed to their legal colleagues, economists are more inclined to approach a problem like that facing us without assuming any particular restriction on existing legal structures. Given a detailed description of the particular characteristics of software, and the objectives sought (e.g. the balance of incentives and access) an appropriate protection structure would be described, and only then perhaps compared with what currently exists. Legal scholars, on the other hand, are more inclined to look at current protection structures together with the outlying creation, and look to see how the existing structure can be made good. Economists would tend to favour bolder, more provocative and novel solutions to the problem, with a lower regard for exactly how (or indeed if) the solutions can be instigated in any real-world economy. Now, of course I do not propose to suggest that economists' solutions are always nice in theory but impractical, while the legal profession would suggest more practical but theoretically deficient solutions. But I am suggesting that each group has many valuable lessons to be learnt from the other, and dialogue between them would be

highly advantageous if a workable *and* theoretically sound solution is ever to be found.[7]

Finally then, I would like to conclude by putting forward my own opinion, for whatever it may be worth. As an economist, when I think about the question of IP protection of software, I am willing to address the problem without restraining myself to the historical accident that has led to it being copyright subject matter for at least 30 years. Rather, I look at software for what it is – a necessary input to the efficient functionality of a computer. Now, a computer is still largely a productive tool, rather than simply a machine for reproducing previously created products (mainly visual images and sounds). Thus, software seems to me to be described much more closely as an input to a productive process (or an intermediate good), rather than a work of authorship to be consumed for its own entertainment pleasure. That is, despite the obvious fact that software can be expressed in exactly the same way as almost all copyright subject matter – as an ordered sequence of symbols –, software is not primarily intended for the same purpose as most other copyright materials.[8] Even a computer programme whose only purpose is to allow music to be played on a computer is really no more a copyright good than is the internal gadgetry and wiring of a radio! This all leads me to the conclusion that software looks more like patent subject matter than it does copyright subject matter.[9]

The value of a computer programme is not to be found in its text, but rather in what it commands a computer to do for us. It is a relatively simple matter to write two different programmes (different in the sense that neither contains any given string of symbols of reasonable length that is also present, in exactly the same order, in the other programme) that would achieve exactly the same output (e.g. visual screen displays) for each possible input (e.g. keystrokes). If a user is not able to say which exact programme is being used to relate all possible inputs to outputs (since that relationship is identical for both programmes), then we should validly understand that the two programmes are formally identical, even though they contain different expression. This is, of course, a patent type of argument, under which what is important is a process that solves a problem (a relationship between inputs and outputs, from the users point of view), and when two different processes both solve the same problem in ways that are functionally identical, the typical efficiency arguments behind patent law require that only one should be allowed to survive.

Now, I should point out that there are certainly some aspects of computer programmes that should retain copyright rather than patent protection. Actual screen designs, any characters and logos that appear on screen during the running of the

[7]Exactly this type of interchange was the driving factor behind the organisation of the workshop from which the papers in this symposium are derived.

[8]As an analogy, consider a racing car and a tractor. At a first, and basic, level of description (motorised vehicles, with internal combustion engines and four wheels), they might look similar. But they are used for completely different things, and we would expect that the law would treat them differently.

[9]Of course readers may well object, citing other examples of copyright protected works that are embodied in functional objects (e.g. architechtual plans for buildings). But the visual image of buildings are consumed in the same way as works of authorship. I consider that software is more akin to the design of internal electrical circuitry (which is designed for functionality rather than estetic image) than the plans for the external appearance of a building. And if the design plans for internal wiring are also protected by copyright, then I would suggest that perhaps they should not!

programme, and other such pleasantries are clearly aspects that are purely intended for entertainment rather than pure functionality. These are rather clearly copyright subject matter.

I am certainly not the first to consider that patent alone, or at least something closer to patent than to copyright, might be a more appropriate protection mechanism for copyright. Although it falls short of advocating patent as a protection mechanism for software, at the heart of much of the paper by Samuelson et al. (1994) is an implicit argument that suggests that software is more akin to patentable, rather than copyrightable, subject material.[10] In that path-breaking paper, we find quotes such as "... the primary source of value in a program is its behavior, not its text." (Samuelson et al., 1994, p. 2315), and "... programs are, in fact, machines (entities that bring about useful results, i.e. behavior) that have been constructed in the medium of text (source and object code)." (Sameulson et al., 1994, p. 2316). The question to address is why then do not Samuelson et al. end up advocating patent alone as the best protection mechanism for software. The answer is only because of the innovative step requirement in current patent law; "... patent law requires an inventive advance over the prior art before it grants protection. Protecting incremental innovations in program behavior through patent law would thwart the economic goals of the patent system: to grant exclusive rights only when an innovator has made a substantial contribution to the art and advanced competition to a new level." (Samuelson et al., 1994, p.2346). I agree, but in the end it seems to me that patent needs far less tweaking (clearly we would need to lower the innovative step requirement, and possibly also the duration of protection for the special case of software) than does copyright to accommodate software.

In spite of my opinion that software is closer to patent than copyright, the patent system as it stands would almost certainly not be appropriate. Both the current innovative step requirement and the duration would need to be addressed for patent protection of software. Also, regulators should be careful about interoperability being used to consumers' disadvantage, and so may have to alter the type of patent protection afforded to software from what is given to other types of innovation. So my way of looking at the issue can be seen to be a sort of *sui generis* solution, although it can also be accommodated within a version of the clean-slate approach, but with hardly any input from the current copyright regime.

In keeping with the basic theoretical efficiency underpinnings of the patent system, in exchange for this special type of patent protection, rights holders would be required to divulge source code information on their programme,[11] as parts of it may be useful for other programmers to develop completely different applications.[12] So long as the new application does not compete with the existing (patented) one,

[10]Also, Karjala (1998, p. 43) argues "Copyright protection for most program innovation would be much worse than whatever comes out of patent, because copyright was not designed for, and indeed is ill-suited to, the protection of technology"

[11]Of course, divulging source code when a programme is patented appears to leave it entirely open to copying in its entirety, perhaps rendering the protection useless in practice. But computers only understand object code, not source code, and object code could remain hidden (perhaps using DRM type features). Anyone wanting to reproduce the programme, would firstly have to re-write the object code from the source code, something that most software users would not be able to do anyway.

[12]Full disclosure in return for patent protection is, of course, a more efficient outcome than reverse engineering, a most wasteful proceedure under which resources are spent to rediscover things that already exist.

there is no obvious reason why the current patent holder's welfare would be diminished by the use of a part of his source code in the development of the new application. Of course, arguments can, and would, be voiced based upon one person's efforts in developing code being used gratuitously by others for economic gain, but this is no different from what happens in other innovative processes that are patent protected and divulged. The first innovator would have written his code with the objective that the particular application that he wanted to achieve works properly, and any other use of this code is, in a sense, purely accidental or fortuitous rather than foreseen by the original programmer, and thereby it is not deserving of financial remuneration. If a particular sub-routine or partial string of code within the entire piece of software that leads to the original application were worthwhile as a stand-alone patentable object (i.e. it would have to have proven applicability for the improvement of other programmes, and be clearly defined as the solution to a given problem), then there is no reason why it could not be patented as a separate entity to the application that it leads to. In this case, new applications that make use of this subroutine would, of course, have to license it from the patent holder.

The possibility of patentability of software has lead to a large group of lobby organisations, whose principal argument seems to be that patenting of software would lead to anti-competitive behaviour, involving large corporations stifling small companies (see, for example, the material on the website http://www.nosoftwarepatents.com). However, Mann (2004) looks at the issue of licensing of software patents, and provides information that seems to fly in the face of such arguments. Mann suggests at least two reasons why patenting of software might not lead to anti-competitive behaviour. Firstly, it turns out that most firms do not rely upon patents to establish themselves anyway ("... about 80% of venture-backed software firms do not obtain patents during the early years of their existence."(Mann 2004, abstract)), and that patents are not used strategically as an entry deterrent; "Incumbents ... rarely use patents to exclude smaller firms from the market." (Mann 2004, abstract). Mann's conclusion is that software patents are beneficial to small, not large, software firms, giving them bargaining leverage, signalling their technical competence, and making the firm attractive to potential acquirers. In a nutshell, patents are seem to primarily benefit smaller firms trying to find a foothold from which they can compete. Mann notes that "... existing practices in the industry suggest that technology is readily available, rebutting the prominent claims of a patent 'thicket' that is supposedly stifling innovation in the industry." (Mann 2004, abstract).

5. Conclusion

In this paper, I have taken an exploratory look at the current debate concerning the extension of IP protection of software from its traditional copyright, to include patent as well. There exist a series of arguments, primarily in the legal literature, that advocate different solutions ranging from the total exclusion of patent, through partial inclusion under a *sui generis* regime, to an argument based on the total overhaul of the current copyright-patent system. Economists, however, have had little input to this debate. In lieu of solid economic analysis on the appropriate structure of IP protection for software, I have offered a personal opinion, which is based on the idea that software is far more like an innovative input to productive processes than a work of authorship. That opinion leads to a system that appears,

at first sight, to be rather more provocative than what legal scholars have suggested. Where the legal literature typically suggests either leaving software to be protected only by copyright, or perhaps melding some patent aspects into a copyright framework, I see it more appropriate to start with a basic patent model, and alter some of the details of the parameters of protection, but not always along the lines of copyright. I suggest shortening, rather than lengthening, of basic patent protection for software (i.e. not at all like copyright), but I would also suggest that the innovative step of current patent law be lessened for the particular case of software (i.e. a bit more like copyright).

References

Barwolff, M. (2002), "Beyond Copyright and Patents for Software", mimeo. Available online at http://ig.cs.tu-berlin.de/ma/mb/ap/2002/Baerwolff-Beyond-2002.pdf

Baseman, K.C., F.R. Warren-Boulton and G.A. Woroch (1995), "The Economics of Intellectual Property Protection for Software: The Proper Role for Copyright." Working Paper Industrial Organization Number 9411004, Economic Department of Washington University. Available online at http://ideas.uqam.ca/ideas/data/Papaers/wpawuwpio9411004.html.

Dreyfuss, R.C. (1993), "A Wiseguy's Approach to Information Products: Muscling Copyright and Patent Into a Unitary Theory of Intellectual Property," *Supreme Court Review 1992*; 195-234.

Einhorn, D.A. (1990), "Copyright and Patent Protection for Computer Software: Are They Mutually Exclusive?", *The Journal of Law and Technology*, 265-78. Available online at http://www.idea.piercelaw.edu/articles/30/p265.Einhorn.pdf.

Ginsburg, J. (1994), "Four Reasons and a Paradox: The Manifest Superiority of Copyright Over *sui generis* Protection of Computer Software," *Columbia Law Review*, December, 2559-2307.

Karjala, D. (1998), "The Relative Roles of Patent and Copyright in the Protection of Computer Programs", *John Marshall Journal of Computer and Information Law*, Fall; 41-74.

Klemperer, P. (2004), "America's Patent Protection Has Gone Too Far", *Financial Times*, March 2, 2004.

Mann, R. (2004), "The Myth of the Software Patent Thicket", University of Texas School of Law Working paper. Available online at http://www.utexas.edu/law/academics/centers/clbe/assets/022.pdf.

Reichman, J. (1994), "Legal Hybrids Between the Patent and Copyright Paradigms," *Columbia Law Review*, December, 2432-2558.

Samuelson, P., R. Davis, M. Kapor and J. Reichman (1994), "A Manifesto Concerning the Legal Protection of Computer Programs," *Columbia Law Review*, December, 2308-2429.

Widdison, R. (2000), "Software Patents Pending?", *Journal of Information, Law and Technology*, issue 3 (available online at http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2000_3/widdison/).

Richard Watt; E-mail: richard@serci.org.