Dynamic Inefficiencies of Intellectual Property Rights from an Evolutionary/Problem-Solving Perspective: Some Insights on Computer Software and Reverse Engineering

Luigi Marengo[a] and **Simonetta Vezzoso**[b]

[a] *St. Anna School of Advanced Studies, Pisa, Italy, e-mail: l.marengo@sssup.it*

[b] *Law Department, University of Trento, Trento, Italy, e-mail: svezzoso@economia.unitn.it*

Preliminary version, *June 2006*.
Please, do not quote without permission

Abstract

This interdisciplinary paper focuses on an evolutionary and problem-solving approach to intellectual property rights in order to discuss some controversial issues in the European legislation on computer software and in some recent competition law case (e.g. the Microsoft case).
Given such claims, we argue that a standard "Coasian" approach to property rights, designed to cope with the externalities of semi public goods may not be appropriate for computer software, as it may decrease both ex-ante incentives to innovation and ex-post efficiency of diffusion. On the other hand the institutional definition of property rights may strongly influence the patterns of technological evolution and division of labour in directions which are not necessarily optimal.
Taking the European legislation on computer software and some recent competition law cases as an example, this paper intends to show that a more careful balancing of costs and benefits, both in static and dynamic terms should be suitable for a pro-innovation IP regime and competition policy.

1. *Introduction*

The current legal attitude favouring a strong intellectual property rights regime is underpinned by the economic reasoning that, unless property-like rights for ideas are established, the inventor would not have the necessary incentive “to devote the time and effort required to produce the invention” (Friedman, 1962, who also extends this reasoning to the “writer”). Without IPR’s, others could reap the fruits of the inventor’s intellectual works. This results from two peculiar characteristics of ideas, that is their non-excludability and non-rivarly. Because of non-excludability, the inventor is expected to suffer free-riding problems, whereas non-rivarly causes non optimal rationing. The property right-like protection is therefore a means to facilitate exchange and internalise externalities (Demsetz, 1967).

The most recent economic literature, however, has adopted a much more cautious approach in evaluating the likely impact on innovation of intellectual property rights regimes. From the legal scholarly community, also, many have invoked a thoughtful review of the available protection system because of its negative impact on innovation and creativity. On the legislative side, we can witness a few, albeit still quite hesitant, signs that a revision in the general policy attitude overwhelmingly in favour of a strong legal protection for new ideas may slowly be under way. This is shown, e.g., by the rejection of the European Directive on Computer Implemented Inventions by the European Parliament or by the recent debate on the “interoperability clause” in French copyright law. According to some commentators, there are also some promising signs that U.S. Supreme Court might take a more pro-competitive stance towards patent law in the near future (Reichman, 2006). Moreover, the European Commission’s decision on the Microsoft competition case, currently under judicial revision, takes quite a critical stance towards the Redmond firm’s claim that the refusal to provide to competitors in the workgroup server market interoperability information allegedly covered by various forms of intellectual property rights was justified by the need to protect its investments and, by that, its innovation incentives (Vezzoso, 2006).

Much in line with the now “fashionable” critical approach, in this paper we point to some dynamic inefficiencies of intellectual property rights from an evolutionary/problem-solving perspective. In particular, we contend that the characteristics of the production processes and of learning processes themselves should deserve a much greater attention in the actual debate.

In the following we will elaborate extensively on this approach and then move to the analysis of the legal treatment of reverse engineering of computer programs.

2. An evolutionary/problem-solving approach to intellectual property

The traditional perspective on property rights is a prototypical application of a Coasian perspective. The knowledge generation process of R&D is exposed to the possibility of strong positive externalities that determine underinvestment. The Coasian solution is therefore to define property rights and create markets for the externality. However, in the case of knowledge externalities things are complicated by a few specific factors. First, the monopoly right conferred by any property right has in this case much more serious consequences as by definition innovative knowledge has no (or little) competing fungible knowledge resources. Assigning enforceable IPR's amounts therefore to assigning high and persistent rents. Second, if knowledge is non-rival (but non-rivarly is somehow over-emphasized) then it is somehow paradoxical that the institutional solution to the incentive problem for knowledge production our society has preferred is to introduce an artificial scarcity for a fundamental resource for which the curse of scarcity would not apply.

But there is another and more subtle sense in which the Coasian solution is at least in need of further and deeper investigation, and concerns the issue of the "granularity" of property rights (Marengo *et al.* 2000). In principle, the Coasian perspective would imply that property should be granted and enforced separately on every – however "small" – right with economic value, and in some sense this route has been followed by the theory and practice of IPR's in the last decades, when, as well documented in a wide array of empirical studies, the domain of (technological) knowledge has been so finely divided by property claims - on essentially complementary pieces of information - that the cost of reassembling constituent rights in order to engage in further research seems to be charging a heavy burden on technological advance. In the realm of scientific and technological research, this has taken the form of a spiral of overlapping patent claims in the hands of different owners reaching ever further upstream. This attitude towards fragmentation is much in line with a Coasian effort to create as many rights as there are markets. Actually, the co-extensiveness of markets and property rights is hailed by economic wisdom as the setting in which competition can promote efficiency at its best: no transaction costs coupled with well defined and perfectly exchangeable property rights lead to perfect allocative efficiency.

Ideally, in a perfectly Coasian world a market would exist for every right with an economic value (Coase 1960). This presupposes individual property rights to be perfectly (and costlessly) defined, perfectly (and costlessly) enforced and perfectly (and costlessly) exchangeable. In this way every inefficient allocation would be avoided. In turn, the whole argument presupposes the very possibility of a limitless separability of property rights and of an ever finer definition thereof. National authorities, both in the US and the EU, have adopted an attitude towards patenting that clearly reflects these principles. As a matter of fact, IP rights are being granted on increasingly fragmented "chunks" of knowledge such as single genes, databases, algorithms or parts thereof. However, as suggested by Coase himself, the finest possible property rights structure is very likely to induce less rather than more competition as underlying markets will be so thin, with respect to the number of agents involved, as to induce monopolistic behaviours and considerable transaction costs. It thus seems that, in the end, allocative inefficiencies might arise which are not less serious than those which a strong and fine-grained rights structure was meant to eliminate. Indeed, a fast growing literature, both in the economic and legal discipline, is currently debating and questioning the idea that "more property rights imply more efficiency" and the idea that "commons, however defined and practiced are tragic".

In this respect, we suggest that the problem-solving nature of the innovative process implies that more property rights do not necessarily generate more innovation and that moreover the institutional arrangements of rights do not only impinge upon the speed of innovation but also upon its direction. From a theoretical point of view, the Coasian argument is based upon the (often only implicit) assumption that technology is given, technological separable interfaces are pre-defined by the state of technological knowledge and property rights (and transaction costs) only determine the efficiency of different coordination modes. This assumption is very clearly expressed by Williamson, who begins his analysis imagining that "in the origin there were markets", i.e. that markets would – if it wasn't for transactions costs – always provide the most efficient mode of allocating resources.

As argued by some critics (cf. Dow, 1987, Granovetter, 1985, Bowles, 1985, Marengo and Dosi 2005), both theoretical arguments and empirical evidence run strongly against this "neutrality" assumption and contend that institutional arrangements and technologies co-evolve and property rights and coordination modes do influence the pace and direction of technological change. In particular, more finely defined intellectual property rights may cause sub-optimal technological trajectories. More in general we submit that the efficiency and sustainability of resource coordination systems crucially depend on technological characteristics of resources themselves and on contractual and institutional patterns of their usage in precise historical moments.

There can be little contention that the production of new software presents the typical characteristics of problem-solving activities. Notably, it is a process of designing viable solutions in a huge combinatorial problem space, characterized by very diffused interdependencies. As pointed out by Simon (1969), problem-solving by boundedly rational agents must necessarily proceed by decomposing any large, complex and intractable problem into smaller sub-problems that can be solved independently, by promoting what could be called the division of problem-solving labour. At the same time, note that the extent of the division of problem-solving labour is limited by the existence of interdependencies. If sub-problem decomposition separates interdependent elements, then solving each sub-problem independently does not allow overall optimization. In this respect, an innovation, e.g. in software, is the outcome of the search in large combinatorial spaces of components that must be closely coordinated. A first important consequence – which emerges very clearly in software design – is that decompositions are mere conjectures which allow to tackle complex problems but at the same time pose limiting constraint to the kind of solutions which can then be developed within the given decomposition. Because of these limits and constraint having a variety of heterogeneous decompositions and architectures has in general positive consequences in terms of the evolutionary properties of the system.

Solving (technological) problems usually requires the complex coordination of many elements or components which present strong interdependencies (i.e. modifying one component has broad consequences on the performance of many other components), which are – if anything for complexity reasons – only partly understood and grasped by problem solvers. Thus the search space of a problem typically presents many local optima and marginal contributions of components can rapidly switch from negative to positive values, depending on which value is assumed by the other components. For instance, adding a more powerful engine could lower the performance and the reliability of an aircraft if other components are not simultaneously adapted (Vincenti, 1991). In a chess game, a notionally optimal strategy could involve, for example, castling at a given moment in the development of the game but the same castling as a part of some sub-optimal (but effective) strategy could turn out to be a losing move. Finally, introducing some routines, practices or incentive schemes that have proven superior in a given organizational context could prove harmful in a context where other elements are not appropriately co-adapted. As a consequence, in the presence of strong interdependencies, one cannot optimize a system by separately optimizing each element it is made of.

It is important to remark that the introduction of any decentralized interaction mechanism, such as a competitive market for each component does not, in general, solve the problem in an optimal way. For instance, if we assume that each component is traded in a competitive market, superior components might never be selected. Thus, interdependencies undermine the effectiveness of the selection process as a device for adaptive optimization and they introduce forms of path-dependency with lock-in into sub-optimal states that do not originate from the frictions and costs connected to the selection mechanism, but from the internal complexity of the entities undergoing selection.

As Simon (1969) pointed out, an optimal decomposition (i.e. a decomposition that divides into separate sub-problems all and only the elements that are independent from each other) can only be designed by someone who has a perfect knowledge of the problem (including its optimal solution). On the contrary, boundedly rational agents will normally be forced to design near-decompositions, that is decompositions that try to put together, within the same sub-problem, only those components whose interdependencies are (or, we shall add, agents believe to be) more important for the overall system performance. However, near-decompositions involve a fundamental trade-off: on the one hand, finer decompositions exploit the advantages of decentralized local adaptation, that is, the use of a selection mechanism for achieving coordination “for free” together with parallelism and adaptation speed. But on the other hand, finer decompositions imply a higher probability that interdependent components are separated into different sub-problems and therefore cannot, in general, be optimally adjusted together. As shown in Marengo and Dosi (2005) and Marengo *et al.* (2000), in domains of highly interdependent entities, such as complementary patents, there are delicate trade-offs between the exploitation of the advantages of decentralization and the need to control for complex interdependencies and that optimal dynamic search path usually are not generated by highly fragmented structures.

One way of expressing the limits that interdependencies pose to the division of problem-solving labour is that market performance signals are not able to effectively drive decentralized search in the problem space. Local moves in the "right direction" might well decrease the overall performance if some other elements are not properly tuned. As Simon puts it, since an entity (e.g. an organism in biology or an organization in economics) only receives feedback from the environment concerning the fitness of the whole entity, only under conditions of near independence can the usual selection processes work successfully for complex systems (Simon 2002). A further aspect concerns the property that, in general, the search space of a problem is not given exogenously, but is constructed by individuals and organizations as a subjective representation of the problem itself and through the very process of problem-solving which defines a focal framework for future representations. If the division of problem-solving labour is limited by interdependencies, the structure of interdependencies itself depends on how the problem is framed by problem-solvers. Sometimes problem-solvers make major leaps forward by reframing the same problem in a novel way. As shown by many case studies, major innovations often appear when various elements that were well-known are recombined and put together under a different perspective. Indeed, one can go as far as to say that it is the representation of a problem that determines its purported difficulty and that one of the fundamental functions of organizations is precisely to implement collective representations of the problems they face.

As we have briefly recalled above, the economic reasoning underlying the current policy attitude towards intellectual property rights can be held to mainly reflect the Coasian view on economic organisation. Admittedly, the definition of many markets or IPR's could provide, under certain circumstances, a good solution to the innovation problem (but perhaps only if opportunism, agency and other problems are neglected). In fact, the higher the degree of decentralization the smaller the portion of the search space and therefore the speed and the directions of the decentralized local adaptation. However, near decomposability can be an exceedingly powerful architecture for effective organization of a complex problem like innovation in many sectors. This is likely to yield technological trajectories that are not optimal. In more formal terms, if the entities under selection are made of many components which are interacting in a complex way, the resulting selection landscape will present many local optima (Kaufman 1993) and selection forces will be unlikely to drive such entities to the global optima. The higher the degree of decentralization the smaller the portion of the search space which is explored and the lower therefore the probability that the optimal solutions are included in such a small portion of space. Sub-optimality and diversity of organizational structures can persistently survive in spite of strong selection forces (Levinthal 1997). Sub-optimality is due to the persistence of inferior features which cannot be selected out because of their tight connections with other favourable features. In other words, whenever the entities under selection have some complex internal structure, the power of selective pressure is limited by the laws governing internal structures.

The main point here is that the incentives or market solution for innovation is most likely to produce less technological efficiency. The evolutionary/problem-solving approach is also able to show that there are quite delicate trade-offs between different levels of suboptimality in the achieved solution and different speeds of adaptation. Careful consideration of the characteristics of production processes and of learning processes would therefore seem to suggest that market-like decentralized mechanisms do not provide appropriate signals in this early problem-solving/innovation phase. As search proceeds and a local peak (a set of standards in the techno-organizational design problem) is selected, the degree of decentralization can be greatly increased in order to allow for fast climbing of this peak (and indeed transaction costs factors can very well be responsible at this stage for variations of the degree of integration), but the more decentralization is pushed forward, the more
unlikely it will be that new and better local optima can be discovered. This would seem to involve the important policy consideration that the system should allow for a more flexible coordination of interdependent elements ("pieces of knowledge") than the Coasian market solution would allow for. In the next Section we will elaborate on these insights taking the rules of reverse engineering of computer software as an example.

3. An application: the case of reverse engineering of computer software

3.1 The IP protection of computer programs in a nutshell

IPR's norms can be seen as the most relevant rules of the game providing for the institutional structure of innovation systems. Among IPR's regimes of different states or intergovernmental organisations (like the European Community as far as e.g. community trade marks are concerned), there still exist substantial differences, and this in spite of policy moves towards globally harmonized standards of protection. Moreover, the levels of exclusivity accorded by the various intellectual property titles vary significantly.

On the one end of the spectrum we possibly have the patent system. The conditions to be met in order to enjoy patent protection are somehow more stringent than those for other forms of IPR's. In Europe, for example, an invention may enjoy protection from patent law provided it is novel, it is susceptible of industrial application, and involves an inventive step. Patent protection grants that the results of individual search for solutions become "proprietary" and cannot be freely used by others. More precisely, a patent covers the inventive idea as such (as expressed in the patent claims defining the matter for which the protection in sought in terms of technical features), and can be invoked even against independent inventors of the same idea. Further, not only a third party's devise (or process) that falls within the literal scope of a patent claim infringes the patent, but also a devise "equivalent" to the claimed invention may face infringing liability. Improvements of the invention and follow-on innovations can be actively prevented by the patent owner (or his patentees), and this in spite of the fact that most leading patent regimes contain experimental use or research exemptions, albeit of a very narrow extent. As part of the patent "bargain," the would-be patentee must disclose the specifications of his work. This requirement aims at giving the public technological know-how, and it is obviously particularly important when inventions are not easily reverse-engineered, or, in economic terms, are partly *excludable* ideas (whereas, for inventions like e.g. pharmaceuticals, bringing the product to the market as such normally implies disclosure). Moreover, most patent laws foresee provisions for compulsory licensing, but they apply only under very strict conditions. Finally, in patent law there are no further explicit "fair use" provisions.

Starting from the 1960s, U.S. courts had been holding that software was not eligible for patent protection, since it was considered similar to abstract ideas and laws of nature. Since the 1981 *Diamond v. Diehr* landmark decision, however, both the U.S. courts and the U.S. Patent Office gradually broadened the scope of protection granted to software-related inventions,. Today, Courts apply the so called "useful, concrete and tangible result" test to software patent claims, and do not require the operation of the program to affect the physical world in any tangible way. As seen above, the would-be patentee is required by law to disclose the specifications of his work. Unfortunately, however, in the case of computer software this requirement prove scarcely effective, since where the patentability of computer software is admitted, as in the U.S., there is no obligation to disclose the implementing source code of the program. Finally, since in patent law there are no further explicit "fair use" provisions, reverse engineering through decompiling would most probably be judged illegal under patent law, and certainly so in case of pure software claims. In fact, the very moment a patented program is run on a computer for decompilation, a object code copy of the program in the RAM memory is realized which could be interpreted as an infringement (Cohen and Lemley 2001).

In Europe, the Convention on the granting of European patents (EPC in the following) expressly excludes patent protection for computer programs (non-patentability because of the subject matter). Therefore, for pure software to be protected on the basis of patents, the exclusion under Article 52 EPC should have to be removed. Computer implemented inventions (e.g. where a devise is controlled by embedded processors - microchips), though, have increasingly been granted patent protection by the European Patent Office (which issues "bundles" of national patents). This evolution within Europe has been marked by some landmark cases. In *Koch & Sterzel/X-ray*, for instance, the Board of Appeals ruled that "..if the program controls the operation of a conventional general-purpose computer so as technically to alter its functioning, the unit consisting of program and computer combined may be a patentable invention". Moreover, the Board recognized that "a claim directed to a technical process carried out under the control of a program (whether implemented in hardware or in software) cannot be regarded as relating to a computer program as such within the meaning of Article 52 EPC". Admittedly, making a distinction between software eligible for patents under European law and software that is not cannot be regarded as an easy task. With this problem in mind, the "technical contribution requirement" was put at the core of the 2002 European Commission's proposal of a directive on the patentability of computer-implemented inventions, which was then rebutted by the European Parliament. As many commentators have stressed, however, the notion of "technicality" put forth by the Commission would have hardly improved legal certainty.

On the opposite end of the spectrum, we encounter the "open movement" in connection with computer programs ("open source") and now increasingly artistic works ("open content", mostly associated with the Creative Commons project). In case of open source software, the idea is also by itself proprietary but it is freely available to others in the form of source code. Users of open source are typically permitted to use, copy, distribute and modify the software, but only as long as they abide by the rules of the open source licence agreements they have agreed to (General Public License, Lesser General Public License, MIT License, etc.). Interestingly, there are significant differences among open source license agreements. Some require for instance the obligation by the licensee to release and license back to the open source community all modifications of the source code, whereas others permit modifications of the code to be kept proprietary under certain conditions.

Moreover, software owners can rely on trade secrets law for the protection of their intellectual works. This further property-like right lasts as long as computer program's unique character remains confidential. Reverse engineering is permitted without limitations. But the "weakness" of a trade secret is that it vanishes once it is broken. Of course, the risk of a trade secret being lost is particularly high if the software is licensed.

The last, and somehow more "traditional" and pervasive form of IP for software is copyright protection. According to Article 10 of TRIPS (Agreement on Trade-Related Aspects of International Property Rights), computer programs, whether in source or object code, are protected as literary works provided that they are original and tangible. Under Article 9, copyright protects the actual code of the computer program, and the way the instructions have been drawn up, but not the idea underlying them. Therefore, as well as for patents, the solutions covered by copyright law cannot be freely used; but copyright protects not the idea as such but its "original expression". Thus, for example, copyright covers the particular form of a statuette used as a lamp base, but not the idea of using a statuette of a human figure as a lamp base. In similar terms, for what concerns computer software, source code and object code are typically protected against literal copying, but not the program's ultimate function. The idea underlying a given software could then be used to "build" a different program. Moreover, and differently from patent protection, no disclosure obligation is foreseen.

Copyright is the form of software's IP protection we will deal with in more detail. In particular, our main focus in the following will be on the legal treatment of the reverse engineering of computer programs. First of all, we develop an economic problem-solving perspective on reverse engineering, and then we will take a closer look at the relevant rules of law, especially following European Directive 91/250 on the legal protection of computer programs.

3.2 On the economics of reverse-engineering from an evolutionary/problem-solving perspective

Reverse engineering is widely diffused in computer software, but usually not for the purpose of making a copy of the good itself, as it is often the case in reverse engineering of physical artefacts. On the one hand, in fact, in order to make a perfect copy of a piece of software one does not have to understand anything of its internal structure and functioning, but simply to duplicate the executable files. On the other hand, decompiling and fully reverse engineering a computer program in the form of an executable or object file is usually more difficult, time consuming and expensive than writing from scratch a new program which performs similar tasks.

Reverse engineering in physical artefacts is normally carried out with the purpose of acquiring information on know-how (how the good is designed, how it works, how specific technical problems are solved, etc.) which is not available in codified version. Computer software is indeed a “perfect” codification of its own design, functions and operations, but if the source code is not made public, the object or executable codes are not understandable by a human reader. This obvious observation gives an interesting perspective on the much debated issue of the legal treatment of reverse engineering. If we expressly consider copyright law, this debate seems somehow paradoxical in its very existence. Copyright has always been applied to artefacts whose inner structure was largely transparent and easily grasped by any customer. When reading a book the very act of consuming the good is nothing but a form of reverse engineering, in which the inner structure and the technical solution employed become perfectly transparent at least to an experienced and cultivated reader. On the same token, for consumers having sufficient technical skills, listening to a musical composition or looking at a painting or sculpture reveals all the secrets of its creation. An important consequence is that consumers can freely learn from copyrighted material, re-use it and re-combine it with elements of other (copyrighted or non-copyrighted) material in order to produce new ideas and materials or simply to enhance own education, skills and competencies. J.S. Bach’s sublime art of counterpoint is made of a series of technical solutions to composition problems, which have been freely studied, acquired, re-used, adapted to their own needs by the following generations of composers. In a sense all the cumulative process of culture has been based upon the free reverse engineering of (possibly copyrighted) creations of other people’s minds. Compiled computer code is an exception and probably the first one in history, as it is indeed a totally codified piece of knowledge but the code is not understandable by even an expert human reader. The process of building knowledge upon knowledge which is normally allowed under copyright regime is therefore hindered in the case of closed source software.

All in all, reverse engineering of computer programs is not something the average user would aim to, already quite satisfied if the program runs smoothly on his hardware. Reverse engineering is a matter for specific categories of users, such as those whose peculiar needs are not fully accommodated by the purchased product, and for the broader category of software developers. Most reverse engineering in software is actually carried out with the purpose of learning: learning programming techniques, learning solutions to specific technical problems, learning the functioning of specific routines in order to modify them for user's customized needs, learning, especially, the functional specifications necessary to make a compatible program. The issues therefore should not be whether and, possibly, to which extent reverse engineering is legal, but whether copyrighting something written in a language which keeps its content hidden to customers should be allowed.

In spite of the obvious knowledge-creating and diffusing characteristic of reverse engineering, in a Coasian world this activity would hardly find a safe place. The individual (or collective) discovery attempts should be viewed as acts of free-riding in an otherwise well functioning IPR market. A party interested in knowing the inner secrets of an item for some commercial purpose which could prejudice the economic interests of the IP rights owner should attempt to obtain that information by way of licence. In fact, thanks to the licensing system, the IP owner recovers its research and development costs. If bypassing licensing by way of reverse engineering were allowed, according to this view, there would not be other means of finding out what is efficient and what rights should therefore be enforced. Unless some clear source of market failure is identified, the IP owner's interest to license and by that reap the economic benefits from any market use of its ideas should not be interfered with. Only uses that could not have been reasonably licensed anyway should be free for use.

On the contrary, we suggest that reverse engineering is not in itself a threat to a firm's incentives to produce innovative software, as this threat comes rather from the so-called (almost) infinite expansibility (David, 1992), i.e. the possibility to copy digital goods in form of computer files very cheaply. It is rather a threat to a firm's strategy to keep a proprietary position on interoperability standards, i.e. on the interfaces between platforms, operating systems and application software which play a crucial role in competition. Designing computer software is in fact a prototypical example of a complex quasi-decomposable problem: designers make conjectural decompositions of the hugely complex problem and establish communication protocols between the quasi-separated components. Such communication protocols are sometimes key interfaces for interoperability as they constitute the fundamental links between different parts of an otherwise unitary system.

From a firm’s perspective the choice between making such protocols freely available or not is a difficult one. If protocols are freely available, the system becomes open, and it is very likely that other producers will start developing compatible software and more customers will be attracted by the system. But, on the other hand, the firm itself may loose control of its initial advantage and be exposed to too much competition on all the parts of the system. A vivid example is represented by IBM, which opted for an open system. From the system’s point of view this strategy was certainly successful as IBM’s model of pc has come to dominate the market, but it was not such a success from IBM’s point of view, as the latter quickly lost its competitive advantage both in the hardware market against specialized and more cost efficient component producers and in the software market against Microsoft, which on the contrary tried to keep the system as closed as possible. It must be also pointed out that because of various well known technological and market reasons, in the meantime the most profitable part of the market has progressively moved from hardware to software.

Also closed interfaces, however, may be detrimental from the firm’s perspective as they may hinder the diffusion of the system if there exist competing ones which, thanks to openness, offer a wider range of applications and higher opportunity to respond to specific customers’ needs.

From a social welfare perspective, instead, the main question is whether developers of closed systems would choose not to develop them if forced to reveal interface standards, or to tolerate individual or collective attempts to gather the relevant information. As we have previously argued, the issue is not the danger to have expensively developed programs copied and reproduced at almost zero cost, but to have independent producers entering the market with compatible alternative programs. The question does not have an easy answer, but the empirical evidence which shows the widespread co-existence of open and closed systems providing similar services seems to suggest that closeness is not a *sine qua non* condition.

From a problem solving perspective there is little doubt that the inevitable inefficiencies determined by conjecturally decomposing systems which present strong and widespread interdependencies, should at least be compensated by the advantages of introducing competitive markets for each quasi-separated component because, if the architecture is kept under a strict monopoly, there is no room for adaptation driven by competition. Moreover, as discussed above, closed source codes hinder a fundamental collective learning mechanism and is bound to reduce knowledge cumulation.

Therefore, from an evolutionary/problem-solving perspective, the act of reverse engineering would seem to acquire quite a different significance. As we have already pointed out, interdependencies among "chunks of knowledge" such as distinctive pieces of software are not known in advance to anyone but make out a substantial part of the trial-and-error activities conducted by agents working on complex problems. Despite the fact that these attempts are mostly of a "commercial" nature and also potentially able to prejudice the economic interests of those who can already claim IP rights on "chunks of knowledge", their social value per se can hardly be denied. Given the complexity of the problem, it seems, a collective search strategy quite opposite to the "market partitioning" philosophy would seem a more efficient solution from a technological point of view. In this respect, it is not evident that reverse engineering is a free taking of somebody's else legitimate rewards for his innovative efforts. On the contrary, it resembles more a collective innovative activity to discover interdependencies among elements that form a complex system.

3.3 An evolutionary/problem solving preliminary assessment of the rules on reverse engineering of computer programs in the EU

All modern IPR's regimes provide for some levers tempering the IP owner's assertion of the exclusive right to use "his own" ideas. They seek to draw a balance between, on the one hand, providing an incentive for the creation of works desired by society and, on the other hand, insuring enough freedom ("public domain") to permit later comers to build on the existing intellectual achievements for further technological and cultural advances. Much in line with this fundamental trade-off, the European Directive 91/250 on the legal protection of computer programs (in the following European Software Directive), aiming at the harmonization of the laws of member States of the EU, contains a so-called decompilation exception for purposes of developing interoperable programs.

Similarly, some key decisions by U.S. courts explicitly considered, and allowed, reverse engineering for that purpose. Thus, it has been stated by that reverse engineering object code to discern the unprotectable ideas in a computer programme is fair use. Thus, *Sega v. Accolade* involved computer games producers. Sega was at the time producing the Genesis console and had a leading position in the market of both consoles and games also thanks to its policy of not revealing the specifications necessary to have non-Sega produced or licensed games running on the same console. Accolade did exactly that: by reverse engineering they acquired such specifications and started selling Accolade produced games running on Sega consoles. The Ninth Circuit affirmed that "[D]isassembly of copyrighted object code is, as a matter of law, a fair use of the copyrighted work if such disassembly provides the only means of access to those elements of the code that are protected by copyright and the copier has a legitimate reason for seeking such access".

Whereas the U.S. courts rely on the broad fair-use provision in copyright law to assert the lawfulness of reverse engineering, in the European Union a specific rule is provided, stating that copyright in a computer program, under certain circumstances, is not infringed by decompilation activities. Thus, under Art. 6(1) of the European Software Directive, the authorization of the IP rightholder is not required where reproduction of the code and translation of its form are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program. This decompilation exception is enforced by the provision of Art.9(1) that anti-decompilation clauses in software contracts are null and void. One of the preliminary issues here is to understand what it is exactly meant under interoperability and the Directive defines it "as the ability to exchange information and mutually to use the information which has been exchanged". Interoperability under the Software Directive, therefore, refers to the ability of a software to exchange information (communicate) with another software (*software-to software* interoperability).

However, under the European Software Directive, some additional conditions should be met, among which that the information necessary to achieve interoperability has not previously been readily available to the person decompiling the program and that the acts of decompilation are confined to the parts of the original program which are necessary to achieve interoperability. Moreover, the information obtained thanks to decompilation should not be used for goals other than to achieve the interoperability of the independently created computer program and should not be given to others, except when necessary for the interoperability of the independently created computer program. Finally, that information should not be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.

Apparently, the interoperability provision under the European Software Directive was the result of a compromise, aiming at accommodating substantially diverging interests. Thus, for instance, major U.S. computer companies had lobbied intensively not to have a decompilation-for-interoperability exception inserted in the final text of the Council Directive. At least some commentators and interested parties would agree that it has worked rather well since its adoption, almost 15 years ago. Indeed, a recent European Commission’s Report concludes that "the [Software]Directive and in particular the decompilation provisions were the result of intensive debate among all interested circles and the balance found then appears to be still valid today" (EC Commission, 2004).

From an evolutionary/problem-solving perspective on reverse engineering, however, it could be argued that the decompilation clause under the European Software Directive is too narrowly crafted. In fact, it would seem to neglect some essential characteristics of the production and learning processes we briefly recalled in the previous Sections. In particular, interdependencies among distinctive pieces of software very often are not known in advance to anyone but are gradually discovered thanks to trial-and-error activities conducted by agents. From a technological point of view, this collective search strategy to discover interdependencies among elements that form a complex system has an indisputable value, that should be reflected on the legal treatment of reverse engineering. In this respect, the statutory limitation to only those which are actually creating compatible product, would seem to create a heavy and unnecessary burden on the parties seeking the application of the decompilation provision. This becomes even more apparent considering that there is no research-exception, or error correction-exception for decompilation. Instead, *Sega v. Accolade*, mentioned above, states that there may be other legitimate purposes for decompilation.

Moreover, the European Directive limits follow-on uses that can be made of information obtained in the course of decompilation. The decompiler cannot publish information learned during reverse engineering. In fact, Article 6(2) of the Directive prevents the publication or trafficking in information by those who have decompiled existing programs. Decompilers are therefore prevented from diffusing the information they learned in the course of their reverse engineering efforts, by that at least partly recouping their expenses From an evolutionary/problem solving perspective, it seems, there could be sound reasons to rebut this ample restriction on the uses the decompiler can make of the information acquired.

A more general question, raised in the context of the European *Microsoft* case under competition law, is whether a reverse engineering exception should in most cases be regarded as appropriate, or, under certain circumstances, an obligation to disclose interoperability information should be imposed. As the European Commission itself affirmed in its Microsoft Europe decision “(D)epending on the size of the program to be decompiled, reverse-engineering may be very costly and time-consuming, and there may also be technical barriers”. Moreover, through legitimate actions such as upgrading the software, compatibility could easily be broken. Therefore, the Commission concluded that “reverse engineering . as opposed to disclosures from Microsoft, does not constitute a viable solution for companies willing to compete with Microsoft on the work group server operating system market” (paras. 36 and 683-687 ff.). The Commission is certainly correct in pointing out the difficulties, uncertainties and costs of reverse engineering. From the evolutionary/problem solving perspective, however, a broadly framed decompilation provision, fostering not only the competitors’ individual but also collective discovery endeavours would possibly seem more adequate.

Finally, even more serious threats to the reverse engineering of software under copyright law could derive from the legal reinforcement of the technical measures to protect digital content. The underlying, alleged reason for outlawing circumvention measures was that, otherwise, copyrighted works provided digitally would have been too vulnerable to infringements The 1996 WIPO Copyright Treaty foresees that the member States should provide “adequate protection” and “effective remedies” against circumvention of technical protection measures, and both the U.S. and the European legislators have made it illegal to circumvent those measures. Already, there have been a few attempts by firms in the U.S. to prevent their competitors from circumventing technological measures in order to reverse engineer products. Moreover, at the European level, some advocate the introduction for computer programs of anti-circumvention provisions, with the possible effect of preventing or restricting the application of the decompilation-for-interoperability exception provided for in the European Software Directive.

4. Preliminary conclusions

The production of new software presents the typical characteristics of problem-solving activities. It is based upon conjectural sub-problem decompositions, whereby a decomposition establishes an overall design which on the one allow the search for solution of sub-problems of manageable complexity, along with division of labour and specialization, but on the other hand puts clear limits on the type of solution which can be produced within that given architecture and almost inevitably generates inefficiencies and bottlenecks in the search process. In addition, the production of new software displays strong cumulativeness, mutual learning effects, re-usability, and, being part of a complex system, suffers from the interoperability problem.

In this paper we have argued that a traditional Coasian perspective on intellectual property rights does not give a satisfactory account of these peculiarities, as it is entirely focused upon the solution of a purported externality "problem". We could say on the contrary that in problem-solving technologies externalities are not a problem but a part of the solution. In particular, we have suggested that reverse engineering is not usually carried out in computer software with the purpose of merely producing cheaper copies of other people's innovative products, but with the purpose of learning and ensuring interoperability.

From a legal point of view, the evolutionary/problem-solving perspective could possibly provide distinctive arguments supporting a clear IP provision in favour of reverse engineering (see e.g. Samuelson and Scotchmer 2002 and Cohen and Lemley 2001 for other potential benefits of reverse engineering). Of course, as far as software is covered by patent law, this branch of law should foresee the possibility to lawfully decompile other parties' software, not only to achieve interoperability but also for (re)search, albeit commercial, purposes. Where a reverse engineering exception is expressly provided for, as in the case of Art. 6 of the European Software Directive, this, it seems, should be framed in much broader terms than the current legal rule.

References

Bowles, S.: 1985, The production process in a competitive economy: Neo-hobbesian and marxian models, *American Economic Review* vol .75, 16-36.

Coase, R.H. (1960), The problem of social cost, in: Journal of Law and Economics, Vol.3, pp. 1–44

Cohen, J. E. and Lemley, Mark A. (2001), Patent Scope and Innovation in the Software Industry, in: 89 *California Law Review* 1, pp.1-58.

David, P.A.(1992), Knowledge, property, and the system dynamics of technological change, in: *Proceedings of the World Bank Annual Conference on Development Economics*, pp. 215–248.

Demsetz, H. (1967), Toward a theory of property rights, in: American Economic Review, Vol. 57, May, pp. 347-359.

Dow, G. K.: 1987, The function of authority in transaction-cost economics, *Journal of Economic Behavior and Organization* vol. 8, 13-38.

EC Commission (2004), Commission Staff Working Paper on the Review of the EC Legal Framework in the Field of Copyright and Related Rights.

Farrell, J. and Salomer, G. (1985), Standardization, Compatibility, and Innovation, in: Rand Journal of Economics, Vol. 16, pp. 70-83

Friedman, Milton (1962), Capitalism and Freedom, University of Chicago Press.

Granovetter, M.: 1985, Economic action and social structure: The problem of embeddedness, *American Journal of Sociology* vol. 91, 481-510.

Levinthal, D.: 1997, Adaptation on rugged landscapes, *Management Science* Vol. 43, 934-950.

Marengo, L. and G. Dosi (2005), “Division of Labor, Organizational Coordination and Market Mechanisms in Collective Problem-Solving”, *Journal of Economic Behavior and Organization*, vol. 58, 2005, pp. 303-326.

Marengo L., Pasquali C. and Valente M. (2005), “Decomposability and modularity of economic interactions”, in W. Callebaut and D. Rasskin-Gutman (eds.), *Modularity: Understanding the Development and Evolution of Complex Natural Systems*, The Vienna Series in Theoretical Biology, Cambridge MA, MIT Press, pp. 835-897.

Reichman, J.H. (2006), Patent Law Harmonization and the Draft SPLT, Paper presented to the World Intellectual Property Organization’s (WIPO) Open Forum on the Draft Substantive Patent Law Treaty (SPLT), Geneva, Switzerland, 1-3 March.

Samuelson, P. and Scotchmer, S. (2002), The Law & Economics of Reverse Engineering, in: 111 Yale Law Journal, pp. 1575-1663.

Vezzoso, S. (2006), The Incentives Balance Test in the EU Microsoft Case: A Pro-Innovation "Economics-Based" Approach?", in: European Competition Law Review, Vol. 7, pp. 382-390.