The Journal of World Intellectual Property (2006) Vol. 9, no. 4, pp. 426–444

Software Patenting

Georgios I. Zekos

Patentable inventions in computer science cannot be whole software programs. An executable software program stored in the memory of a computer is an example of a machine component, embodied in electrical signals, having a particular physical structure. Software components could be patented within tangible inventions; however, pure software is still largely unprotected. Examiners are required to look, not at how the computer interprets the algorithm, but at the results of the process. The determination of the patentability of the subject matter must be made on the basis of externally observable behavior of the device in question, rather than on any knowledge of whether a device is implemented in hardware or in software. A software invention can be a new physical or electronic and virtual object, which performs a new function, either electronic/virtual or physical, with virtual effects felt in the real world.

Keywords software; patentability; cyberspace; computers; algorithm

1. Legal background

The computer industry has overlooked software in favor of more valuable computer hardware.[1] A computer program, or software, is a "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result".[2] Software patent refers to any patented innovations that can be embodied in software. Is software alone patentable? It is argued that software patents are inappropriate because software is a cumulative technology, proceeding by sequential innovation, and so a software invention characteristically builds on tens or hundreds of previous innovations. Moreover, it is said that patent "thickets" will inevitably impede the flow of new software products.[3]

Inventors will be encouraged by the temporary monopoly provided by a patent enhancing their likelihood of profitable exploitation.[4] Software patents do not normally occupy a large product space and a patent protects a distinctive feature or set of features of the software. Patentable inventions in computer science are not whole software programs but individual ideas or approaches to particular problems.[5]

To get a patent, the applicant must make a disclosure of the invention specific enough that the invention can be reproduced by a person with normal ability in the appropriate art (either under license or free of cost when the patent has expired). Open-source software is an exception, as the source code is fully disclosed. The requirement for disclosure in software patents is set too low to ensure the simple re-creation of an invention.[6] Patents allow the refinement, practical application, and deployment of inventions that might otherwise fade away because of the inventor's lack of funds, entrepreneurial zest, or an external agency. The Bayh-Dole Act of 1980 gave universities the patent rights to inventions arising from government-funded research, facilitating the spread of innovations by conventional commercial channels rather than merely through academic publications and unpublished reports.

The patent laws reflect the effort to balance the rights granted to inventors against the costs of granting those rights.[7] New technologies have often presented problems for the patent and copyright systems.[8] Patents were designed for new and useful inventions, whereas copyrights were for original expressive works.

Computer programs are, on the one hand, utilitarian works, not so different in some ways from the apparently patentable widget. On the other hand, they are expressive written works, which have traditionally been protected by copyright law. Computer programs initially fell into an ambiguous space in the intellectual property system, and the existing doctrines and definitions were ultimately incapable of effectively channeling software into one form of protection.[9] Useful mathematical algorithms fell within the ambit of patentable subject matter.[10] The Federal Circuit's change of heart concerning patentability did not include a change in the "copyrightability" of computer programs.

The European Patent Convention (EPC) draws distinction by excluding software programs from patent protection unless they have a "technical effect". US law demands that a software program should have a "practical utility" to constitute patentable subject matter.

The aim of this article is the examination of the patentability of software.

2. What is software?

The term "software" is used to refer alternatively to algorithms and procedures carried out by software programs, descriptions of software written in a natural language, source code written on paper or stored in a computer's memory, and executable software stored on a persistent storage medium such as a hard disk.[11]

Executable software programs are physical components of computers and so software programs are characterized as "intangible", "abstract ideas", and "mental processes", to differentiate them from physical hardware. The intangibility of the processes performed by a computer does not render the engine itself intangible. It could be said that an executable software program is not itself "intangible" merely because the processes it performs are intangible or because the information it conveys is intangible. Electrical signals are tangible (physical) in the sense that they are subject to the laws of physics and, like matter, can act upon and be acted upon by other energies and matters, and so electrical signals are not "intangible" in the same sense as ideas, mental processes, or mathematical formula, which are not subject to the laws of physics.[12] Thus, an executable software program stored in the memory of a computer is an example of a machine component embodied in electrical signals having a particular physical structure. For example, Microsoft

Word resides in the memory of a computer in the form of stored electrical signals and so, regardless of the fact that Microsoft Word, as a physical structure, is invisible, it exists in the form of a particular configuration of electromagnetic fields, which are physical in the sense just described. Hence, different executable software programs have different physical structures that interact with the computer hardware in inimitable ways to cause the computer to perform specific functions. A computer includes both hardware and software, and the architecture of modern digital computers includes both hardware and software components. The software components may be modified and in this sense are "soft".

More specifically, the modern computer architecture includes a processor, a memory, and a set of instructions stored in the memory. The processor retrieves instructions from the memory and executes them. The processor and memory are fixed and so they are hardware. The set of instructions is also referred to as a "program" or, more generally, as "software". Different programs may be stored in the same memory and executed on the same processor. The hardware serves as a *platform* on which software may be executed.[13] The underlying distinction between software and conventional electromechanical devices is the exclusive process by which software is designed and instantiated.

Invention refers to the process of designing the physical structure of a new and useful machine, and so computers have automated the process of invention. It could be argued that executable software programs that perform useful functions are electromechanical machine components and so software programs should be susceptible to protection by patent law, because patent law is the branch of intellectual property law that protects machines and machine components. Designing executable software programs exclusively in terms of their logical structure raises problems for patent law because patent law has limited patent protection to physical products and to physical processes that have practical utility. The scope of a product patent claim is limited to the physical structures that the patent enables the public to make and use, and the scope of a process patent claim is limited to the specific steps of the process and any recited physical structure upon which such steps function.

Patent law's utility requirement requires *practical* utility.[14] Conception has been said to be the "touchstone of inventor-ship, the completion of the mental part of invention". The mere identification of a problem to be solved, a result to be achieved, or a function to be performed does not satisfy the conception requirement if the conception of specific physical means for solving the problem, achieving the result, or performing the function is absent.[15] The mere mental formation of an idea of a mathematical formula or scientific theory, absent the mental formulation of a specific practical application of the formula or theory, does not satisfy the conception requirement.

Software patentability has focused on whether software qualifies as statutory subject matter and courts have attempted to apply patent law's physicality requirements directly to software, mainly by premising subject-matter patentability on the mere presence or absence of physical structural terms in software patent claims and specifications.[16] It is argued that claims that include ‘‘physical’’ terms (such as ‘‘machine’’ or ‘‘memory’’) will satisfy the statutory subject-matter requirement, while claims lacking such magic words will fail to satisfy the requirement. The lack of physical terms would show that the inventor had not yet invented an individual physical product or process, but perhaps had only devised a simple ‘‘wish’’ or ‘‘plan’’ for a product or process. The lack of physical terms in a software claim does not by itself designate that the inventor has not invented a software program capable of being embodied in a physical form or that the inventor is attempting to claim more than she/he has invented. A software claim may solely use logical terms to point out the useful, novel, and non-obvious features of the claimed program. It is not even *possible* for a programmer to conceive of or describe a program in terms of its physical structure and in that way satisfy the formal physicality requirement. Moreover, describing a program in terms of its physical structure would not increase the public’s ability to make and use, or even to understand, the invention.[17] The only option available to software inventors is to describe their inventions in logical terms because it is not possible to describe them in physical terms. It could be said that the applicant can draft the claims in terms of the physical hardware that performs the steps of the algorithm, and the result will be a ‘‘new machine’’, and so a software invention may satisfy the statutory subject-matter requirement simply by claiming the invention in terms of a functionally equivalent hardware implementation. The simple fact that a software patent claim is written in entirely functional language does not imply any malicious intent on the part of the claim drafter, but reflects an attempt to claim a software invention in the only terms in which it is reasonably susceptible to being claimed. Patent law has long allowed product inventions that are not susceptible to being claimed in terms of their physical structure to be claimed in other terms, such as in terms of the process by which the invention is made, and so patent protection should not be denied to an invention purely because it is not susceptible to description in conventional terms.[18]

The mere absence of ‘‘physical’’ terms in the description and claims of a software patent is not a trustworthy indicator that the claims are not directed to a statutory subject matter resulting in false negatives and so a finding that specific computer programs are not patentable subject matter, regardless of the fact that the programs themselves are embodied in a physical form and perform useful functions. Specific computer programs are a statutory subject matter simply because they are embodied in a physical form and described in physical terms, even though they perform no useful function. Phonographs’ record novelty might lie solely in the originality of the song recorded on it. The question of software patent claim scope is predominantly significant, not only because of the technological importance of software and the prospective economic impact of overbroad or underbroad software patent claims, but also because software patent claims tend to be written using functional language. In the framework of software patents, patentability seems to be more a game of all-or-nothing than one in which claim scope is corresponding with the scope of invention, and so a software claim written in purely functional terms encompasses *all* physical entities that perform the claimed function. The novel, non-obvious, and valuable features of a software program may be described clearly and with particularity without using physical terms. Software programs, nonetheless, are components of machines and intellectual property protection should be available for software designs for the same reasons as it is available for conventional electromechanical designs.

3. Patentability of software

Generally speaking, in order for something to be eligible for patent protection, it must be useful (as required under 35 U.S.C. § 101); it must be novel (as detailed under 35 U.S.C. § 102); it must be "non-obvious" (as detailed under 35 U.S.C. § 103); and it must fit both a written description and enabling requirement (as detailed under 35 U.S.C. § 112).

It could be argued that patents block the entry of newcomers into the software field. The question has to be asked whether marketed software packages are clones of flourishing products, with insignificant originality. Could cloning a patented product be a patentable invention? Cloning a successful product could be patentable if there is an invention making the practical usage of the new product more efficient, but the innovator must devise an original solution that does not infringe on another's patent by just copying the whole patented product with reproduction.

When the business was established, patents protected research and development investments against appropriation by free-riders and enabled innovations to be traded with competitors. At the beginning, IBM considered that computer programs and procedures were not patentable.[19] In 1965, it was held that programs could not be patented because the US patent system unambiguously excluded mathematical laws (and, consequently, computer algorithms) as patentable subject matter. In 1964, the US Copyright Office held that programs could be afforded copyright protection under a "rule of doubt", provided that a human-readable copy of the program source code was deposited in the office,[20] enabling a competitor to inspect the code of the program and reverse engineer a clone.[21] In 1968, Goetz applied for a patent taking advantage of an August 1966 advisory by the US Patent and Trade Mark Office (USPTO) that "a patent could be granted to a program if it could meet the requirements of either a 'process' or an 'apparatus' ". Consequently, the Autoflow program was presented as a machine for achieving a particular kind of information transformation. The Autoflow "machine" transformed source card decks into printed charts. Because of the disclosure required by patent law, competitors were able to study Autoflow, understand its algorithms, advance or design around them, and produce competing products that the market could accept or reject. It could be argued that the workability of the claimed patent, combined with the virtual nature of software, is an unqualified bar to the development of any functionally equivalent product.[22]

Through the 1970s, software was considered equivalent to mathematical algorithms or laws of nature, and thus was not patentable.[23] The United States, following the principles of rewarding inventors and perpetuating the industrial tradition, have expanded the scope of patentability to software. Not all "inventions" are patentable in the United States because neither a mathematical formula nor an algorithm for making mathematical computations or conversions can be patented.[24] The US Supreme Court clarified that computer technology should not be treated in a different way from other technologies under the patent law and that inventions involving a computer program can be a suitable subject matter for protection by US patents.[25] Moreover, in *In re Alappat* it was held that software has the power to switch a general purpose computer into a special purpose machine, which constitutes patentable subject matter.[26] Customarily, software has been protected by copyright and excluded from patent protection in Europe[27] but the European Patent Office is responsible for granting at least 20,000 software patents. According to article 10(1) of the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS),[28] "computer programs, whether in source or object code, shall be protected as literary works", and article 27(1) suggests that computer programs are as patentable as any other product or process. Since the 1970s, software in Europe has been protected mainly by copyright. It has to be taken into account that even if software met the technological criteria for patent protection, such protection would be unproductive because much of computer technology has a very short market life. The claim to copyright in a computer program is based upon a minimum amount of original authorship in the expression or form of the statements or instructions contained in the computer program, and a work is deemed original if it is the autonomous creation of an author. Copyright protection unsurprisingly extends to the human-readable source code of the software. In the United States, the Copyright Act provides protection for software programs so long as they are fixed in any tangible medium of expression.[29] Customarily, the computer industry has relied heavily on trade secret and contract law protection against piracy. Moreover, trade secrets can be more efficient than patents for the protection of software for the reason that many computer programs, which simply adapt an old program to a new use, may be unpatentable as they would be apparent to a computer programmer of ordinary skill in that art. However, patent protection is not disturbed by independent conception or reverse engineering of the claimed invention by a third party at a later date and so patenting is a better way of protecting software because it prevents copying as well as independent creation, which is a defense to a charge of copyright infringement.

The issue of whether or not a computer algorithm constituted patentable subject matter was addressed both in *Gottschalk v Benson*[30] in 1972 and *Diamond v Diehr* in 1981. In *Gottschalk v Benson*, the algorithm could have been executed by any general purpose computer and the patent was rejected by the examiner because the algorithm constituted "non-statutory subject matter". The Supreme Court ruled out "pure" software patents but left the door open for software-enabled inventions that produced a new, useful, and non-obvious technical effect. Many patents were granted for physical inventions that incorporated algorithms and programs, and the patent specifications included complete program listings and it can hardly be disputed that these incorporated algorithms. The incorporation of embedded microprocessors was so widely accepted as an incremental product improvement that these innovations were accepted by the Patent Office but it was difficult to draw a line where a patent would be rejected on the grounds of non-statuary subject matter. Hence, software components could be patented within tangible inventions, while pure software was still largely unprotected. The number of software patents being granted is evidence enough that the threshold of obviousness is too low. Establishing novelty in software patents is also problematic. In *Diamond v Diehr*, the Supreme Court ruled that the invention was so specific that it did not rule out the use of the algorithm in other contexts.

The last 25 years of software innovation has enjoyed patent protection and, since 1996, the USPTO in fact has had guidelines for dealing with software patents.[31]

Copyright protection was designed to protect only the *expression* of work, not its function legitimizing the efforts of competitors of best-selling products to reverse engineer the program and create a competing clone product. Reverse engineering did not require the direct copying of code, but rather created a working replica by the observation of the program's black-box behavior.[32] For example, Lotus sought to apply the obscure concept of look-and-feel infringement and the District Court of Boston[33] stated that Paperback Software had infringed Lotus' copyright by copying the menu structure of 1–2–3, implying that user interfaces would have to be notably dissimilar to avoid infringement, whereas users of competing products benefited by having a common user interface across applications. Thus, it was acceptable to imitate an interface or menus to assist in product switching, but the cloned interface should not be the key interface of a program.[34]

The President's Commission on the Patent System proposed that computer programs be denied patent protection.[35] The debate surrounding software patents in the United States has focused not on the question of whether a specific software innovation meets the statutory requirements of patentability, but rather on the question of whether software innovations are the kind that is protectable *at all* under federal patent law.[36] It was considered that software innovations were not patentable subject matter because software innovations are not a process, machine, manufacture, nor composition of matter and so the Supreme Court initially declared software innovations unpatentable.

In *Gottschalk v Benson*,[37] the court ruled that the software innovation in question was not protectable under patent law owing to a lack of patentable subject matter and so, in declining to characterize an algorithm as protectable subject matter, the court relied on two fundamental precepts: (1) that an abstract idea, a law of nature, or a natural phenomenon is not patentable subject matter; and (2) that a patent must have a definitive scope. The decision in *Benson* interprets an algorithm, like a mathematical formula, to be equivalent to a law of nature and thus not of patentable subject matter.

Patents on abstract ideas, laws of nature, and natural phenomena are not patentable subject matter, making a distinction between mere discovery of that which already exists in nature and true invention.[38] Claims lacking definitive scope are unpatentable. In *Diamond v Chakrabarty*,[39] the Supreme Court overruled a USPTO ruling and found that man-made micro-organisms were patentable, thereby, providing patent protection for ''anything under the sun that is made by man''. An algorithm is patentable subject matter so long as the algorithm only uses mathematical relationships to achieve its ends, as opposed to trying to patent the mathematical relationships themselves and so software, at least under certain circumstances, was a patentable subject matter.[40] An algorithm will not be patentable subject matter if it amounts to nothing more than an attempt to claim ''math'', as mathematics falls outside the scope of patentable subject matter. Where a mathematical formula is used as part of a process, even where the formula is well known and, so long as the process itself is otherwise patentable, a patent may be issued. An *application* of a law of nature or mathematical formula to a known structure or process may well be deserving of patent protection. Courts developed the Freeman–Walter–Abele test[41] to determine whether an algorithm was patentable or whether it was merely a non-patentable mathematical process. The USPTO or a court determines whether a mathematical algorithm is recited directly or indirectly by a claim. If recited directly, the second portion of the test is to determine whether the claim *is directed to* the algorithm (in which case it is not patentable subject matter) or rather the claim merely *uses* the algorithm as part of its claimed process.[42] Presently, examiners are directed to make certain that the ''claimed invention as a whole must accomplish a practical application''.[43]

It was considered that software did not *do* anything other than perform mathematics, albeit very quickly and efficiently, and so software was not deemed patentable subject matter. Under the USPTO Guidelines any claimed invention, computer-related or otherwise, must meet the requirements of 35 U.S.C. § 101, including the requirements of utility, and the requirements of novelty and obviousness, as well as meeting certain enabling and written description requirements. Utility of a computer-related invention must be something more than just data or information stored on a computer-readable medium. Examiners are required to look not at how the computer interprets the algorithm but at the results of the process. Where the patent contains both process and apparatus claims the invention is a general-function computer (i.e. the apparatus) that is running certain software (i.e. processes that are contained in the software). Computers[44] ''divide the brains of the operation (the algorithm) from the brawn (the hardware)''. A computer program can be claimed as a pure process patent only in the instance where the ''computer is executing the computer program's instructions''.[45] The problem is not one of patentability, or whether a given invention meets the requirements of novelty, non-obviousness, utility, and enablement, but of whether something qualifies as an invention.

In *Northern Telecom, Inc. v Datapoint Corp.*,[46] the patent claimed an improved method of entering, verifying, and storing (or ''batching'') data with a special data entry terminal and the core of the claimed invention was the combination of components or steps, rather than the details of the program the applicant actually used. Moreover, in *Fonar v General Electric*,[47] descriptions of software are ''satisfied by a disclosure of the functions of the software'' because ''writing code for such software is within the skill of the art'' once functions are disclosed and so there is no need to disclose source code or detailed flow charts, only function. Furthermore, in *Dann v Johnston*,[48] the court held a patent on a ''machine system for automatic record-keeping of bank checks and deposits'' invalid for obviousness focusing on whether analogous systems to the patentee had been implemented in computers before, rather than analyzing the precise differences between the patentee's program and the prior art programs.

Finally, software is better suited to the protective mechanisms of patent than those of the alternative copyright and in *State Street Bank*[49] and *AT&T v Excel*[50] the court unreservedly admitted software to the canon of patentable subject matter. It has to be taken into account that for any given software invention, the remaining statutory criteria for patentability must be worked out on a case-by-case basis, including the requirement that the invention be novel, non-obvious, it be a significant technological advance, and that it complies with statutory disclosure requirements.

4. US versus EPC and EU

Software ''patentability'' focused on the threshold ''subject matter patentability'' requirement, specifying the kinds of subject matter that are susceptible to patent protection. The patentable subject matter requirement is an attempt to give a mechanism for filtering out claimed subject matter that does not fall within the ''useful'' arts (in the United States) or the ''industrial'' arts (in Europe).[51] Processes, machines, articles of manufacture, and compositions of matter are susceptible to patent protection in US law and ''industrial application'', rather than a list of specified categories, is the primary test for patentable subject matter in EPC, providing a list of excluded categories. There is a lack of objective definitions of conditions determining whether any specific software patent claim defines patentable subject matter. Although a computer program that simply performs a mathematical calculation lacking any practical application does not qualify as a patentable subject matter, software that controls an automobile-manufacturing robot probably qualifies as a patentable subject, leaving a substantial grey area in between. A subject matter must achieve a ''useful, concrete, and tangible result'' in order to qualify as patentable subject matter.[52] Both the ''technical effect'' and the ''practical utility'' requirements allow for the exclusion of abstract ideas, laws of nature, and natural phenomena as patentable subject matter. Every computer program executing on a computer produces a ''technical'' effect in the sense that execution of the program causes the computer to store, modify, and transmit electrical signals internally. The subject matter patentability determination must be made on the basis of externally observable behavior of a device in question, rather than on any knowledge of whether a device is implemented in hardware or in software. The software-implemented device would not qualify as statutory subject matter when its execution on the computer purely involves the manipulation of electrical signals inherent to the execution of any computer program. Practical utility needs something beyond mere internal physicality or an abstract idea of calculation.[53] Physical utility, logical utility and application utility are the basis of evaluating software patent claims. A process qualifies as patentable subject matter if it transforms something "into a different state or thing".[54] *In re Pardo*,[55] a process claim directed to controlling the internal operations of a programmed computer constituted statutory subject matter because the claim was directed to "executing programs in a computer", which the court viewed as indistinct from a strictly mechanical adding machine. Besides, an algorithm that failed to perform "physical steps" did not qualify as statutory subject matter.[56] Transformation of data refers to a transformation of physical electrical signals and so a machine that performs a *physical* transformation (e.g. of electrical signals from one form into another) qualifies as a statutory subject matter if the result of the transformation is useful, concrete, and tangible (i.e. physical).

The coordination and control of the internal communications between programs and data files stored in different computers connected as nodes in a telecommunications network is a patentable subject-matter requirement because the invention was concerned with the internal workings of computer processors.[57] Moreover, in *Vicom/Computer-Related Invention*,[58] it was held that:

> if a mathematical method is used in a technical process, that process is carried out on a physical entity (which may be a material object but equally an image stored as an electrical signal) by some technical means implementing the method and provides as its result a certain change in that entity.

The manner in which a mathematical algorithm processed data is used to evaluate if a claimed method is patentable, at least partly on the basis of its logical utility,[59] and so in *IBM/Document Abstracting and Retrieving*[60] the electrical signals generated when implementing the claim do not represent a physical thing. In *AT&T Corp. v Excel Communications, Inc.*, the Court of Appeals for the Federal Circuit eliminated physicality as a requirement for subject-matter patentability. The elimination of the physicality requirement leaves no clear basis for excluding from subject-matter patentability either abstract ideas or a variety of works falling exclusively within the liberal arts (such as solely human-performed innovations in law, politics, and business). Consequently, there is lack of clarity about whether the patentable subject-matter requirement requires a claimed product or process essentially to transform physical material into a different state or thing, or whether it is satisfactory that the product or process in question manipulates data *representing* physical material. Processes that both *perform* and *simulate* physical activity satisfy the patentable subject-matter requirement, without distinguishing between these two senses of physicality or explaining the relevance of either to the subject-matter patentability determination.[61] Patentable subject matter is determined on the basis of whether the data used by the claimed processes *represent* physical entities.[62] Besides, in *In re Schrader*,[63] process claims are not to be directed to statutory subject-matter because they ‘‘do not reflect any transformation or conversion of subject-matter *representative of or constituting* physical activity or objects’’. Hence, there is confusion as to whether a claimed process has to be implemented in a physical activity or objects, or merely to perform operations that are ‘‘representative of’’ a physical activity or objects.[64]

An invention must benefit both its direct users and indirect beneficiaries (e.g. regulatory authorities), irrespective of the specific physical form in which an invention is instantiated or the particular information processing steps performed by the invention. The physical utility takes the ‘‘external perspective’’, while logical utility and application utility arguably take the ‘‘internal perspective’’ both expressed within the functional practicability of the invention. It could be argued that the software must be susceptible to embodiment in a tangible form or intangible form with a tangibility expressed functionally ensuring the practical usage by the public. Moreover, practical implementations of the intelligence of software must be constitutive of material outcomes. Software has to perform functions that were never performed beforehand by machines. Software makes it possible for functions formerly classified within the *liberal arts* to be performed entirely automatically by machines.

An invention has to be susceptible to industrial application, new, and must involve an inventive step, and an invention is considered to be new if it does not form part of the state of the art and is an inventive step if, with regard to the state of the art, it is not obvious to a person skilled in the art.[65] A proposal for an EU directive on software patents indicated that software, in order to be patentable, must show an inventive step, the invention must make a ‘‘technical contribution’’,[66] and, consequently, the ‘‘technical contribution’’ requirement could be satisfied by just running the program on a computer, contributing to the state of the art in a field of technology. The conditions for patentability established in Europe apply to an invention of a technical nature, which is not obvious to a person skilled in the art, and so makes a ‘‘new contribution to the state of the art’’ having ‘‘industrial applicability’’. Software has to be a technical object or a (manufacturing) process in the material world, as opposed to the immaterial world of theories and ideas. Software invention involves the use of a computer, computer network, or other programmable devices. The current practice in Europe is not to allow patenting of computer programs or any software as such, and only to grant patents for inventions that use computer programs in their implementation if an inventive technical contribution is present. Inventions making use of algorithms may be patentable if they are concerned with the solution of a technical problem not monopolizing the algorithm itself or its use in other contexts.[67] The aim of the directive is to harmonize, rather than to change, the legal position, by clarifying the legal framework applying to software.

5. Conclusions

Rather than producing broad innovations to advance the software industry, like inventions such as the word processor, spreadsheet, or presentation graphics, information is being sliced and diced to the point that every trivial combination or extension of prior software technology is being accorded the same protection. The software industry is producing thousands upon thousands of inherently meaningless software patents of dubious value, each a potential threat to innovation and competition. It is argued that software patents are not inducing innovation,[68] which must not be the case in healthy software patenting.

An executable software program may be a part of an electromechanical machine, and executable software programs differ from other kinds of electromechanical machine components because of the distinctive process by which they are designed and instantiated.[69] Executable software is the first kind of electromechanical machine component that may effectively be designed only in terms of its logical structure. Patent law's rules of patentability and claim scope require electromechanical machine components to be conceived of, described, and claimed in terms of their physical structure, not simply in terms of their logical structure. The traditional requirements for patentability (e.g. novelty, non-obviousness, and utility) have to be applied to the logical structure of the claimed software program.

Does software fit into existing intellectual property paradigms without first developing a clear, detailed, and precise conception of what software is, how it works, and how it is created?

Software has unique properties that are relevant to the form of intellectual property protection that should be given to software.[70] While patent[71] law protects ''functional'' products and processes, copyright law protects ''expressive'' works[72] and so patent law is the appropriate means for protecting software if software is solely (or perhaps primarily) functional,[73] and copyright law is the appropriate means for protecting software if software is solely (or perhaps primarily) expressive.[74] Hence, if software is both functional and expressive, then software is susceptible to protection by both patent and copyright law.[75]

The author considers that software will be patentable as long as there is an invention, and the included creation must not be merely science, but it must include spirit expressed materially or be an intangible content expressing functional effects. Cyberspace and new technology creates software that could attribute to innovation, functionality, and inventiveness of new functions, which are more electronic than mechanical with traditional understanding, and so the intangibility of expression of software should not be an obstacle in patentability, as long as there is an invention expressed electronically in a virtual world with real world expressions and effects.

A software invention can be a new physical or electronic and virtual object, which performs a new function, either electronic/virtual or physical, with virtual effects felt in the real world. If the invention is a process which generates electricity, then we define the invention as the new function that is identifiable by a new physical or physically understandable virtual object.

About the Author

Georgios I. Zekos, B.Sc. (Econ.), J.D., LL.M., Ph.D., Attorney at law, Economist; e-mail: zekosg@yahoo.com

Notes

1 R. Trudeau, Software patents, C.I.P.R., **9** (1992), 234.

2 17 U.S.C. § 101 (2000). J.-P. Smets, *Stimulating Competition and Innovation in the Information Society—Patent or Sui Generis Right: What Protection Should be Considered for Software and Other Intangible Innovations?* (March 23, 2001), at ⟨http://www.pro-innovation.org⟩ (Europe's positive law forbids patents on computer programs.)

3 B. Pfaffenberger, The coming software patents crisis: can Linux survive?, *Linux Journal*, (1999), at ⟨http://www.linuxjournal.com/article/5079⟩.

4 G. Zekos, Developments on business method patents, *Journal of World Intellectual Property*, **693** (2004), at ⟨http://www.wernerpubl.com⟩.

5 M.A. Lemley and D.W. O'Brien, Encouraging software reuse, *Stanford Law Review*, **49** (1997), 255, at 295.

6 T.P. Burke, Software patent protection: debugging the current system, *Notre Dame Law Review*, **69** (1994), 1115, at 1158.

7 *Graham v John Deere Co*. 383 U.S. 1, 10–11 (1966) (''The difficulty of formulating conditions for patentability was heightened by the generality of the constitutional grant and the statutes implementing it, together with the underlying policy of the patent system that 'the things which are worth to the public the embarrassment of an exclusive patent,' as Jefferson put it, must outweigh the restrictive effect of the limited patent monopoly.''). J.E. Cohen and M.A. Lemley, Patent scope and innovation in the software industry, *California Law Review*, **89**(1) (2001), n. 5 (''The extent to which the patent system is actually necessary to induce innovation that would not otherwise occur is an unanswered, and perhaps unanswerable, empirical question.'').

8 *Sony Corp. v Universal City Studios Inc*. 464 U.S. 417, 430 (1984) (''From its beginning, the law of copyright has developed in response to significant changes in technology.'').

9 P. Samuelson *et al*., A manifesto concerning the legal protection of computer programs, *Columbia Law Review*, **94** (1994), 2308.

10 *State St. Bank & Trust v Signature Fin. Servs* 149 F.3d 1368, 1373 (Fed. Cir. 1998).

11 M.A. Paley, Article: A model software petite patent act, *Computer & High Technology Law Journal*, **12** (1996), 301, 319 (''A written claim for an algorithm can be expressed either as a literal or non-literal expression of the algorithm. The literal description of the algorithm is the algorithm itself. Hence, the mere literal description of a software process is the process.'').

12 J. Gleick, Patently absurd, *New York Times*, March 12, 2000 (magazine), at 44, at ⟨http://www.nytimes.com/library/magazine/home/20000312mag-patents.html⟩ (''Patents began in a world of machines and chemical processes—a substantial, tangible, nuts-and-bolts world—but now they have spread across a crucial boundary, into the realm of thought and abstraction.''). Although Gleick later acknowledges that software

programs "*are* machines, in a way", he asserts that "they are machines without substance—incorporeal machines, machines made of imagination and 'logic' and 'bits'; ibid. Jim Warren of Autodesk Inc., testified before Congress that "software is not a gadget. . . . Software is what occurs between stimulus and response, with no physical incarnation other than as representations of binary logic"; *Prepared Testimony and Statement for the Record of Jim Warren Before the Patent and Trademark Office* (January 26–27, 1994), at ⟨http://www.bustpatents.com/autodesk.htm⟩ (September 14, 2003). Warren also recommended that the Patent Office "issue a finding that software . . . implements intellectual processes that have no physical incarnation; processes that are exclusively analytical, intellectual, logical and algorithmic in nature"; ibid. Letter from Prof. D. Knuth, the Patent Office, at ⟨http://lpf.ai.mit.edu/Patents/knuth-to-pto.txt⟩ (September 14, 2003). Prof. D. Knuth, stated his belief in a letter to the Patent Office that "[t]he Patent Office has fulfilled this mission [of serving society by formulating patent law] with respect to aspects of technology that involve concrete laws of physics rather than abstract laws of thought", such as software. J.P. Barlow, The economy of ideas, *Wired*, (March 1994) (arguing that "all the goods of the Information Age [including software] . . . will exist either as pure thought or something very much like thought: voltage conditions darting around the Net at the speed of light, in conditions that one might behold in effect, as glowing pixels or transmitted sounds, but never touch or claim to 'own' in the old sense of the word."). G. Ifrah, *The Universal History of Computing: From the Abacus to the Quantum Computer*, pp. 123–124, John Wiley & Sons, Hoboken, 2001 (claiming that Pascal's invention of the Pascaline mechanical calculating device in 1624 "mark[e]d the final break with the long age of ignorance, superstition and mysticism which above all had stopped the human race from contemplating that certain mental operations could be consigned to material structures made up of mechanical elements, designed to obtain the same results."). *Greenwalt v Stanley Co*., 54 F.2d 195, 196 (3d Cir. 1931) (examples of "intangible, illusory, and nonmaterial things" include "emotional or aesthetic reactions"); *In re Schrader* 22 F.3d 290, 296 n. 12 (Fed. Cir. 1994) (implicitly distinguishing "intangible subject matter" from "physical activity or objects").

13 J.C. Phillips, Note: sui generis intellectual property protection for computer software, *George Washington Law Review*, **60** (1992), 997, 1002 ("In the broadest sense, a computer consists of two elements: hardware and software. Hardware includes the physical embodiment and structures associated with a computer system."). A.G. Isztwan, *Computer Associates International v Altai, Inc.*: protecting the structure of computer software in the second circuit, *Brooklyn Law Review*, **59** (1993), 423, 425 ("Hardware consists of the physical electronic circuits in which the processing ordered by the instructions occurs.").

14 *State St. Bank* supra n. 10, at 1375 ("The question of whether a claim encompasses statutory subject matter should not focus on which of the four categories of subject matter a claim is directed to . . . but rather on the essential characteristics of the subject matter, in particular, its practical utility"); see also *In re Ziegler* 992 F.2d 1197, 1201 (Fed. Cir. 1993) (citing *Cross v Iizuka* 753 F.2d 1040, 1044); *Nelson v Bowler* 626 F.2d 853, 856 (C.C.P.A. 1980) ("Practical utility' is a shorthand way of attributing 'real-world' value to claimed subject matter. In other words, one skilled in the art can use a claimed discovery in a manner which provides some immediate benefit to the public").

Malta v Schulmerich Carillons Inc. 952 F.2d 1320, 1341, n. 5 (Fed. Cir. 1991) ("The patent system is directed to practical utility, not to basic research."). *Hilton Davis Chem. Co. v Warner-Jenkinson Co*. 62 F.3d 1512, 1520 (Fed. Cir. 1995) ("The ability of the public to successfully design around—to use the patent disclosure to design a product or process that does not infringe, but like the claimed invention, is an improvement over the prior art—is one of the important public benefits that justify awarding the patent owner exclusive rights to his invention.").

15 *Fiers v Revel* 984 F.2d 1164, 1169 (Fed. Cir. 1993) ("Conception of a substance claimed per se without reference to a process requires conception of its structure, name, formula, or definitive chemical or physical properties."). *Amgen, Inc. v Chugai Pharmaceutical Co*. 927 F.2d 1200, 1206 (Fed. Cir. 1991). *Mergenthaler v Scudder* 11 App. D.C. 264, 276 (1897). Conception is a mental act. *Burroughs Wellcome Co. v Barr Lab*. 40 F.3d 1223, 1232 (Fed. Cir. 1994) ("For conception, we look not to whether one skilled in the art could have though of the invention, but whether the alleged inventors actually had in their minds the required definite and permanent idea."). *Le Roy v Tatham* 55. U.S. (14 How.) 156, 174–5 (1853) ("The processes used to extract, modify, and concentrate natural agencies, constitute the invention. The elements of the power exist; the invention is not in discovering them, but in applying them to useful objects.").

16 *Valmont Indus., Inc. v Reinke Mfg. Co*. 983 F.2d 1039, 1042 (Fed. Cir. 1993). 35 U.S.C. § 103(a) ("Patentability shall not be negatived by the manner in which the invention was made.").

17 *In re Bernhart* 417 F.2d 1395 (C.C.P.A. 1969) (means-plus-function claim held to be patentable subject matter because of recitation of physical terms such as "computer"). *In re Noll* 545 F.2d 141 (C.C.P.A. 1976) (means-plus-function claims held to be directed to statutory subject matter when the specification recited conventional computer hardware elements in conjunction with a computer program for performing the functions recited in the claims). *In re Taner* 681 F.2d 787 (C.C.P.A. 1982) (method claims held to be directed to more than a mere mathematical algorithm and therefore constitute statutory subject matter where the claims recited application of the method to seismic energy waves and other physical entities). *State St. Bank*, supra n. 10 (means-plus-function claim held to be patentable subject matter because of physical structure imported into claim by reference to specification). *Diehr* 450 U.S. 188 (mathematical equation applied to a process for curing rubber held to be patentable subject matter). *In re Alappat* 33 F.3d 1526, 1577 (Fed. Cir. 1994) (means-plus-function claim held to be patentable subject matter because of physical structure imported into claim by reference to specification). *In re Warmerdam* 33 F.3d 1354 (Fed. Cir. 1994) (affirming rejection of method claims which recited no physical structure and reversing rejection of product claim directed to a "machine"). *In re Lowry* 32 F.3d 1579 (Fed. Cir. 1994) (holding claim directed to a "memory" to be statutory subject matter even though claim defined contents of memory in terms of logical entities). *In re Trovato* 42 F.3d 1376 (Fed. Cir. 1994), vacated by 60 F.3d 807 (Fed. Cir. 1995) (means-plus-function claims which recited invention in terms of functional elements held to be non-statutory subject matter because of specification's failure to recite any corresponding structure). *State St. Bank*, supra n. 10, at 1373 (means-plus-function claim held to be statutory subject matter because of definition of invention in terms of physical components in specification). Cf. *Parker v Flook* 437 U.S. 584, 594 (1978) (holding method for updating an alarm limit on a catalytic chemical

conversion of hydrocarbons not to be statutory subject matter despite reference in claim to ''the catalytic chemical conversion of hydrocarbons''). *In re Christensen* 478 F.2d 1392 (C.C.P.A. 1973) (holding method for determining the porosity of a subsurface formation in situ to be non-statutory subject matter because the claimed invention's sole point of novelty was a mathematical formula). *In re Walter* 618 F.2d 758 (C.C.P.A. 1980) (holding method for cross-correlating electrical signals representing seismic waves not to be directed to statutory subject matter).

18 *Atlantic Thermoplastics, 947 Co. v Faytex Corp*. 974 F.2d 1279, 1283–1284 (Fed. Cir. 1992) (Newman, J., dissenting) (recognizing the long-standing acceptance of product-by-process claims). ''The premise of such claims has been called the Rule of Necessity, for it provides a way of patenting inventions or discoveries whose structure is not sufficiently known or knowable to be described objectively''; ibid., at 1279. *In re Painter* 1891 C.D. 200, 57 O.G. 999 (Comm'r of Pats. 1891) (holding that the right to a patent should not be denied, and that the use of the product-by-process claim format is appropriate, in the case of an invention which ''cannot be properly defined and discriminated from prior art otherwise than by reference to the process of producing it'').

19 P. Samuelson, Benson revisited: the case against patent protection for algorithms and other computer program-related inventions, *Emory Law Journal*, **39** (1990), 1025, at 1038.

20 M.R. Wessel, Legal protection of computer programs, *Harvard Business Review*, (March–April 1965), 97, at 103.

21 Reverse engineering would mean examining the source code in order to understand how it worked, transcribing its data structures and formats, and then writing a program with the same functions; A. Johnson-Laird, Software reverse engineering in the real world, *Dayton Law Review*, **19**U (1994), 843.

22 R. Moy, A case against software patents, *Santa Clara Computer & High Technology Law Journal*, **17** (2000), 67, at 99.

23 Courts have long held that mathematical formulae, algorithms, and laws of nature are not patentable subject matter under the Patent Act; *Gottschalk v Benson* 409 U.S. 63, 71–72 (1972).

24 Ibid.

25 *Diamond v Diehr* 450 U.S. 175, 185 (1981).

26 *In re Allapat*, supra n. 17, at 1545.

27 European Patent Convention of October 5, 1973, at ⟨http://www.european-patent-office.org/legal/epc/e/ar52.html⟩.

28 TRIPS, article 10(1), at ⟨http://www.wto.org/english/tratop_e/trips_e/t_agm3_e.htm⟩.

29 *Accusoft Corp. v Mattel, Inc*. 117 F. Supp. 2d 99, 101 (D. Mass. 2000) (citing *Computer Assocs. Int'l Inc*., 982 F.2d 693, 702 (2d Cir. 1992) (The US Copyright Act protects computer programs as ''literary works'').

30 *Gottschalk v Benson*, supra n. 23; *Diamond v Diehr*, supra n. 25.

31 Examination guidelines for computer-related inventions, *Federal Regulation*, **61** (February 28, 1996), 7478.

32 *Lotus Dev. Corp. v Borland Int'l Inc*. 49 F.3d 807 (1st Cir. 1995); *Lotus Dev. Corp. v Borland Int'l Inc*. 799 F. Supp. 203 (D. Mass. 1992); *Lotus Dev. Corp. v Borland Int'l Inc*. 788 F. Supp. 78 (D. Mass. 1992); *Lotus Dev. Corp. v Paperback Software Int'l* 740 F. Supp. 37 (D. Mass. 1990).

33 *Lotus Dev. Corp. v Paperback Software Int'l*, ibid.
34 *Lotus Dev. Corp. v Borland Int'l Inc*. 516 U.S. 233 (1996).
35 *Diamond v Diehr*, supra n. 25, at 197 (Stevens, J., dissenting).
36 T.P. Burke, Software patent protection: debugging the current system, *Notre Dame Law Review*, **69** (1994), 1115, at 1142 (citing 35 U.S.C. § 101 foundation patent requirements but questioning how, with thousands of software patents already issued, ''the question [of] whether software per se can be patented is still hotly debated.'').
37 *Gottschalk v Benson*, supra n. 23. (''Justice Douglas' opinion in *Gottschalk v Benson* virtually foreclosed the patentability of computer programs.''). *O'Reilly v Morse* 56 U.S. 62, 113 (1854) (holding that a particular claim in Samuel Morse's patent on the telegraph was void as it amounted to a claim on any method that used electric signals as a means of communication, even those which had not yet been invented).
38 *Funk Bros. Seed Co. v Kalo Co*. 333 U.S. 127, 131–132 (1948) (holding that a discovery relating to the qualities of certain strains of naturally occurring bacteria is not patentable). *Rubber-Tip Pencil Co. v Howard* 87 U.S. 498, 507 (1874) (''An idea of itself is not patentable, but a new device by which it may be made practically useful is.'').
39 *Diamond v Chakrabarty* 447 U.S. 303, 309 (1980).
40 *Diamond v Diehr*, supra n. 25.
41 *In re Freeman* 573 F.2d 1237 (C.C.P.A. 1978); *In re Walter*, supra n. 17; and *In re Abele* 684 F.2d 902 (C.C.P.A. 1982).
42 *Arrythmia Research Tech. Inc. v Corazonix Corp*. 958 F.2d 1053, 1058 (Fed. Cir. 1992).
43 *State St. Bank*, supra n. 10, at 1374. Manual of Patent Examining Procedure (MPEP) § 2106.
44 O.S. Kerr, Computers and the patent system: the problem of the second step, *Rutgers Computer & Technology Law Journal*, **28** (2002), 47.
45 MPEP § 2106(IV)(B)(1)(a). *AT&T Corp. v Excel Commc'n Inc*. 172 F.3d 1352 (Fed. Cir. 1999); *In re Alappat*, supra n. 17.
46 908 F.2d 931 (Fed. Cir.), cert. denied, 111 S. Ct. 296 (1990).
47 107 F.3d 1543 (Fed. Cir. 1997).
48 425 U.S. 219 (1976).
49 *State St. Bank*, supra n. 10.
50 *AT&T Corp. v Excel*, supra n. 45. In *In re Zurko* 111 F.3d 887 (Fed. Cir. 1997), the CAFC held that a patented software invention was non-obvious even though each of the elements of the invention could be found in the prior art, where the prior art did not identify the problem to be solved. While *Zurko* certainly demonstrates that some software patents will be held non-obvious, it is a specific holding of rather limited utility to most software patentees. *Kewanee Oil Corp. v Bicron Corp*. 416 U.S. 470, 489 (1974) (referring to the ''federal interest in disclosure'' embodied in the patent laws).
51 In the United States, patentable subject matter is defined in 35 U.S.C. § 101. In the European patent system, patentable subject matter is defined in article 52 of the European Patent Convention.
52 *State St. Bank*, supra n. 10. at 1375.
53 Ibid; the CAFC held that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces ''a useful, concrete and tangible result''—a final share

price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.

54 94 U.S. 780, 788 (1876). *Cochrane v Deener* 94 U.S. 780, 788 (1876) (holding that "[a] process is . . . an act, or a series of acts, performed upon the subject-matter to be transformed and reduced to a different state or thing"). *In re Sherwood* 613 F.2d. 809, 819 ("seismic traces [recited in the claims] are electrical signals from geophones, i.e., *physical* apparitions, or particular patterns of magnetization on magnetic tape, i.e., the pattern of the magnetization being a *physical* manifestation, or a *physical* line on a paper chart"). *In re Walter*, supra n. 17, at 767–68, the CCPA distinguished between inventions in which "the end product . . . is a pure number", and which are non-patentable subject matter, and inventions which "produce a physical thing", which may be patentable subject matter even if the physical thing is represented in numerical form.

55 684 F.2d 912, 916.

56 *In re Grams* 888 F.2d 835, 840.

57 IBM/data processor network T6/83, *European Patent Office Journal*, (1990), 1–2.

58 *European Patent Office Report*, **2** (1987), 74.

59 *AT&T Corp. v Excel*, supra n. 45.

60 T115/85, *European Patent Office Journal*, (1990), 1–2.

61 *In re Taner*, supra n. 17, at 790.

62 *In re Abele*, supra n. 41.

63 22 F.3d 290, 294 (Fed. Cir. 1994).

64 See *Arrhythmia Research Technology v Corazonix Corp*. 958 F.2d 1053.

65 European Patent Convention, articles 54(1) and 56.

66 Proposal for a Directive of the European Parliament and of the Council on the Patentability of Computer-Implemented Inventions, COM (02)92 final, at ⟨http://www2.europarl.eu.int/oeil/file.jsp?id = 219592⟩. S.R. Paulsson, *Patenting Software vs Free Software. What Should the European Union Do?*, briefing paper written for the Policy Department for Economics and Science, DG 2, European Parliament (February 2005), at ⟨http://www.ffii.org/~jmaebe/epecosci0502/SoftwarePatent.pdf⟩. Opinion of the Economic and Social Committee on the "Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions" (COM (2002) 92 final—2002/0047 (COD)), [2003] O.J. C61/154. Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions COM/2002/0092 final—COD 2002/0047 [2002] O.J. 151E/129.

67 Written Question E-0194/03 by T. Villiers (PPE-DE) to the Commission, Software patents, [2003] O.J. 161E/192.

68 J. Bessen and R.M. Hunt, The software patent experiment, Research on Innovation Working Paper, 2004, p. 2, at ⟨http://www.researchoninnovation.org/softpat.pdf⟩.

69 *Webster's New World Computer Dictionary* (9th edn), p. 341, 2001 (defining software as a "computer program or programs, in contrast to the physical equipment on which programs run (hardware))"; *Merriam Webster's Collegiate Dictionary* (10th edn), p. 1117, 1993 (defining software as "something used or associated with and usu. contrasted with hardware", such as "the entire set of programs, procedures, and related documentation associated with a system and esp. a computer system"); *Microsoft*

Computer Dictionary (5th edn), p. 489, 2002 (defining software as ''[c]omputer programs; instructions that make hardware work'').

70 J.R. Goodman *et al*., Toward a fact-based standard for determining whether programmed computers are patentable subject matter: the scientific wisdom of Alappat and ignorance of Trovato, *Journal of Pat. and Trademark Official Society*, **77** (1995), 353 (arguing that the particular manner in which computers are programmed using software distinguishes software from other technologies in a way that is relevant to patent law); see also Symposium*:* Toward a third intellectual property paradigm: a manifesto concerning the legal protection of computer programs, *Columbia Law Review*, **94** (1994), 2308, at 2327 (arguing that computer ''programs are machines that happen to have been constructed in the medium of text'' and therefore differ from other kinds of machines and other kinds of textual works for purposes of intellectual property protection); P. Samuelson Symposium: The Semiconductor Chip Protection Act of 1984 and its lessons: creating a new kind of intellectual property: applying the lessons of the chip law to computer programs, *Minn. Law Review*, **70** (1984), 471, at 663 (arguing that computer program code is functional and therefore should not be subject to copyright protection, but rather should be subject to a new form of intellectual property protection).

71 D.S. Chisum, *Patents*: *A Treatise on the Law of Patentability, Validity and Infringement*, section 4.01, Matthew Bender, New York, 2000.

72 The subject matter copyright law protects is defined in 17 U.S.C. § 102(a) (2002): ''Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression . . .''. 17 U.S.C. § 102(b) expressly excludes from copyright protection, *inter alia*, procedures, processes, systems, and methods of operation, which are within the purview of patent law. Symposium: Toward a third intellectual property paradigm: misappropriation as a third intellectual property paradigm, *Columbia Law Review*, **94** (1994), 2594, at 2597 (''Patent law protects creative but functional invention, while copyright protects creative but nonfunctional authorship.'').

73 V. Chiappetta, Patentability of computer software instruction as an ''article of manufacture'': software as such as the right stuff, *Marshall Journal of Computer & Information*, **17**J (1998), L89, at 143 (arguing that ''[a] proper test for patentability of software related inventions must clearly and consistently draw a line separating claims to software as the specific means for computer system implementation of the contained algorithms/processes (which are patentable subject matter) from those using a software context merely to express and communicate those algorithms/processes (which must be tested on their own merit independently of the software context to determine if they involve patentable subject matter.'').

74 Samuelson, supra n. 70, at 516 (noting that, with one exception, patent and copyright protection were mutually exclusive before the advent of software).

75 A.B. Wagner, Patenting computer science: are computer instruction writings patentable?, *Marshall Journal of Computer & Information*, **17**J (1998), L5 (''Patents and copyrights are mutually exclusive statutory interests with no overlap in 'abstract expression' subject matter.'').