GUIDEBOOK
ON
IP RIGHTS IN SOFTWARE

NYS SCIENCE+TECHNOLOGY
LAW CENTER

NEW YORK STATE OF OPPORTUNITY.
Division of Science, Technology & Innovation

About the IP Rights in Software Guidebook:

This guidebook was compiled by the New York State Science & Technology Law Center to help answer questions about how to best protect intellectual property rights relating to software innovation. The Guidebook walks readers through use of patent, trade secret and copyright protection to maximize the commercialization potential for new software.

About the New York State Science & Technology Law Center:

The New York State Science & Technology Law Center (NYS STLC) has been a leading resource in technology commercialization for nearly a decade. Since its inception, the NYS STLC has assisted with hundreds of commercialization projects across New York State. It was established at the Syracuse University College of Law by Empire State Development's Division of Science, Technology and Innovation (NYSTAR) to facilitate New York State's economic development by leveraging the experience and expertise of law faculty and SU College of Law students to assist New York businesses and institutions in delivering new and emerging technologies to the marketplace.

Advisement:

The information contained in this pamphlet is intended to be an introductory guide. No part of the guidebook, attachments, or related discussions constitutes legal advice or written opinion of counsel. For legal advice, please consult with an attorney.

Any opinions, findings, conclusions, or recommendations expressed are those of the author and do not necessarily reflect the views of the New York State Department of Economic Development.

Research by Chris Horacek, Esq. and Kristian Stefanides, L'20

Table of Contents

Introduction

The advance of computers is marked by innovative improvements in computer hardware, microchips, transistors, and memory storage. These innovations have been protected as typical advances in technology—through the patent system. Software innovations that control computer tasks are more challenging to protect. Software is not a machine or article of manufacture, but a written set of instructions that takes expertise and time to create. Computers include sensors, actuators, monitors, and user interfaces, all of which are controlled by software. The software created for such applications has been recognized as intellectual property, and businesses are keenly interested in protecting it, but the best means of protection are evolving.

Intellectual property rights in technology generally can be protected with patents, copyrights, and trade secrets, or a combination of the three. The challenge is to balance whether the cost of seeking the protection is justified in view of the value it will provide and decide which methods of protection should be utilized. This guidebook provides an overview of how each of the three methods of intellectual property (IP) rights is applied to protecting software, and considerations for determining which to pursue. Trademarks are a fourth type of intellectual property. They protect brands used to identify products and services, and are not covered by this guidebook.

1 Software

There are three aspects of software that are key to understanding how to protect it: (1) the design and sequence of processing functions that are executed by the software program, sometimes referred to as the software architecture, (2) the source code written in high-level programing language that comprises the instructions for carrying out each computation or function that the program performs, and (3) the object code that is machine readable binary

code compiled from the source code and loaded on and run by the computer for which the program was written. Each method of protecting intellectual property rights applies only to certain aspects of software. Generally, patents can be used to protect software architecture, trade secrets can be used to protect source code, and copyrights can be used to protect source code as well as the object code compiled from the source code.

The right to protect inventions and original writings is provided for by the United States Constitution in order to promote science and useful arts and encourage innovation. Federal laws include the US Patent Act, Copyright Act and the Defend Trade Secrets Act of 2016. Trade secrets are also protected under common law and state statutes. Each method of protecting IP rights in software has advantages and disadvantages, and often a combination of two or all three methods will provide the best protection. Patents, which protect the architecture of the software program, are discussed in section III; trade secrets, which represent a cost-effective method for protecting source code and architecture that cannot be easily reverse engineered, are discussed in section IV; and copyright protection for source code and its associated object code is discussed in section V.

2 Patent Protection

Patents are granted by the United States Patent and Trademark Office (USPTO) for fully disclosed inventions that are new, useful, and not obvious to someone skilled in the art. The types of inventions eligible for patent protection are processes, machines, articles of manufacture, compositions of matter, designs, and certain types of plants. There are areas of subject matter that the courts have specifically excluded from eligibility for patent protection. For example, scientific laws, mathematical formulas, abstract ideas, and natural phenomena are not eligible for patent protection. The rationale for excluding these categories is that they are discoveries, rather than inventions, and allowing these general discoveries or tools of invention to be monopolized would discourage rather than

encourage invention.

Applying these principles to software raises an issue. Neither source code nor object code constitutes a process, machine, article of manufacture, or composition of matter. However, the unique sequence of processing functions that are executed by the software program is a recognized type of process, and consequently the workflow or ordered sequence of processing steps carried out by a software program can be patented as a process, if the requirements for patentability are met.

The process consists of the number and order of the groups of specific data processing functions that are performed by the software code. Each group of data processing functions can be thought of as a separate software routine or module, and consequently a software process patent will protect the ordered sequence of processing routines or modules that are specified and claimed in the patent. Although source code and object code are not patentable per se, a software patent will protect any source code and object code that is programmed to carry out a process covered by the claimed process, because by definition the code is designed to execute the patented process. A software patent therefore protects not just one version of source code or object code, but any source code or object code that carries out the process claimed in the patent. This is a key advantage of a software patent over a copyright of software.

When a patent is granted, the patent owner has the exclusive right to exclude others from using software utilizing the patented process, offering it for sale throughout the United States, or importing it into the United States. Patent protection is typically enforceable from issuance until the date 20 years from the filing date. An important consideration is that obtaining a patent does not guarantee an owner the right to make and commercialize products that embody the patented technology without a license. An invention that improves upon or uses preexisting patented technology requires a license of the preexisting patent to avoid infringing the preexisting patent. This is another reason that review of existing patent litera-

ture is very important.

2.1 Patent Protection for Software

As mentioned above, mathematical formulas are not patentable, and this restriction requires evaluation of the patentability of algorithms, in general, versus a process implemented by a computer program. An algorithm can be defined as a formula or well-defined set of computational or processing steps that will solve a specific problem, and generally, algorithms are not patentable. An algorithm can be performed manually, with general computational software, such as a spreadsheet, or by specialized application-specific software. Since an unpatentable algorithm is a set of processing steps and a potentially patentable software process is also a set of processing steps, what is the difference between the two?

An algorithm typically solves one discrete problem or does one computation, and the term applies to any implementation of the algorithm, whether done manually, on a spreadsheet, or with application-specific software. A software program typically employs many algorithms performed by application-specific software in a prescribed sequence to solve a composite problem that requires many computations and processing activities. The individual, application-specific algorithms in a program are often described as software routines or modules, and the potentially patentable process is the specific way in which the routines or modules are sequenced and linked. In other words, the architecture defined by the combination of routines and modules, which is typically depicted as a flow chart, can potentially receive patent protection. Essentially one must demonstrate that the combination of routines and modules can be used to perform a specific function or solve a specific problem for a particular end user.

In considering whether to pursue patent protection, it is important to understand the architecture will be disclosed as part of the patent examination process. Even if ultimately the patent is denied, the application is published and becomes available to competitors. It

is therefore very important to be familiar with the prior art in the field and be able to distinguish the invention from it. This process is somewhat complicated with software because the courts are still considering the type of software patents that should receive patent protection.

2.2 Issues with Patenting Software

Patenting software is an evolving field, and the best information about the type of software patents that will be granted and upheld by the courts comes from decisions the courts have made about contested software patents. An example is the Alice Corp. v. CLS Bank International case decided by the Supreme Court in 2014. An overview of the Alice case follows.

The Alice case involved a computer-implemented method of reducing settlement risk in financial transactions using a third-party intermediary. Settlement risk refers to the risk that only one party will satisfy its obligation under an agreed-upon financial transaction. The patent at issue in the Alice case implemented a computer system to set up shadow accounts to reflect the actual balances of two financial institutions (banks), and thereafter instructed the banks to make permitted transactions based on the account balances, thereby facilitating the exchange of financial obligations.

The Court concluded that "the concept of intermediated settlement is a fundamental economic practice long prevalent in our system of commerce, and the use of a third-party intermediary (or clearing house) is a building block of the modern economy. Thus, intermediated settlement, like hedging, is an abstract idea."

In reaching its determination, the Court laid out a two-part test for determining whether a patent is ineligible, because it is directed to exclude subject matter, such as an abstract idea or an algorithm. First, the Court determines whether the claims of the patent are directed toward one of the judicially excluded subject matters, such as "abstract idea." If so, the Court searches for additional elements

in the claim that transform the excluded subject matter into patentable subject matter. The additional elements claimed must ensure that the patent in practice amounts to significantly more than a patent upon the abstract idea itself. In other words, the claim must include a number of elements that limit the claim to solving a specific problem using application- specific software. The Court stated that "[T]he relevant question is whether the claims . . . do more than simply instruct the practitioner to implement the abstract idea . . . on a generic computer."

Regarding the patent at issue in the Alice case, the Court stated that the function performed by the computer was "purely conventional," and that the computer essentially did no more than keep electronic records. The Court found that the claims did no more than recite the concept of intermediated settlement as performed by a computer, and therefore added "nothing of substance to the underlying abstract idea." Nevertheless, the court held open the potential for software patents to meet patent eligibility criteria.

The Alice case illustrates the challenges associated with obtaining a patent on new software. There remains no bright-line rule for when software is patentable, although the courts continue to grapple with it. While the case lays out an analytical framework, the criteria within the framework are vague and open to interpretation.

The U.S. Patent and Trademark Office (USPTO) has attempted to provide guidance through memos on subject matter eligibility available. Current information can also be found in the Manual of Patent Examining Procedure (MPEP) available on the uspto.gov website.

2.3 USPTO Memos

In June 2014, the USPTO issued a memo to its patent examiners (the 2014 memo). The 2014 memo stated that the Alice Corp. decision neither created a per se excluded subject matter category, such as software or business methods, nor imposed special requirements

for patentability of software. The 2014 memo outlined an analytical framework for patent examiners to follow when considering software and business method patent applications. First, examiners determine whether the claim is directed to the four categories of invention (process, machine, manufacture, or composition of matter). If the claim does not fall within one of the four categories, it must be rejected. If the claim does fall within one of the categories, the examiner must follow the two-part analysis stated in Alice Corp. as outlined above; i.e., if the description of invention subject matter is an abstract idea, does it contain sufficient additional novel and non-obvious elements to transform it into patentable subject matter.

The 2014 memo gives examples of what constitutes an abstract idea: fundamental economic principles; certain methods of organizing human activities; ideas or concepts; and mathematical relationships and formulas. "Fundamental economic principles" include creating contractual relationships, hedging, and mitigating settlement risk. "Certain methods of organizing human activities" include "using an algorithm for determining the optimal number of visits by a business representative to a client," and "computing a price for the sale of a fixed income asset and generating a financial analysis output." "Mathematical relationships and formulas" include mathematical formulas for hedging, and formulas for managing life insurance policies by performing calculations and manipulating the results.

In order to be patent-eligible, claims that include these types of abstract ideas must then be examined to determine whether the idea has been applied in a manner such that the claim amounts to significantly more than the abstract idea itself. Claims that may be enough to qualify as "significantly more" include:

- improvements to another technology or technical field;
- improvements to the functioning of the computer itself;
- meaningful limitations beyond generally linking the use of an abstract idea to a particular technological environment.

Meaningful limitations that can make a software process patentable often take the form of a specific combination of a number of pro-

cessing steps that solves an application-specific problem. However, if a claim does no more than require a computer to perform "generic computer functions that are well-understood, routine and conventional activities previously known to the industry [,]" the claim fails the "significantly more" standard.

In July 2015, the USPTO issued an update regarding subject matter eligibility in the wake of Alice Corp. The 2015 update gives detailed examples and analysis of claims that both meet and fail the "significantly more" criteria. The update also stresses the importance of reading the elements of a claim both separately and in combination to determine whether the claim amounts to significantly more than an abstract idea.

For example, "generic computer components that individually perform merely generic computer functions are able in combination to perform functions that are not generic computer functions and amount to significantly more." However, computer functions that simply perform repetitive calculations, receive, process, and store data, or automate mental tasks have been found by courts to constitute conventional and generic functions, and thus fail the "significantly more" standard.

2.4 Novelty and Non-obviousness

If subject matter eligibility is established, the patent examiner will then consider novelty. To determine whether the invention is new, the examiner will search for "prior art" related to the claimed invention. Prior art includes patents, patent applications, descriptions in a printed publication, or products in public use, on sale, or otherwise available to the public on, at, or before the time a patent application is filed. If the invention is identical to the prior art, then it is deemed "anticipated." Prior art also anticipates the claimed invention if all elements described in the claims of an application are contained in that prior art.

The non-obviousness requirement of the Patent Act assesses

whether an invention would have been obvious to a practitioner skilled in the particular art at the time of filing. The patent examiner assumes a hypothetical person with ordinary skill in the relevant field of technology would be aware of all of the prior art existing at the filing date of the invention for purposes of determining non-obviousness. The Supreme Court has established various tests for determining if an invention is obvious. In 2007, the Supreme Court announced a new multi-factor test to evaluate non-obviousness in its KSR Int'l Co. v. Teleflex Inc. decision. The test focuses on six factors, in part designed to distinguish the creative characteristics of someone of ordinary skill in the art that are obvious from those that are inventive. The idea is that ordinary creativity would be obvious to a person of ordinary skill, and ordinary creativity is not enough to satisfy the non-obviousness requirement. The following are the six factors:

1. Combining prior art elements according to known methods to yield predictable results;
2. Simple substitution of one known element for another to obtain predictable results;
3. Use of a known technique to improve similar devices, methods, or products in the same way;
4. Applying a known technique to a known device, method, or product ready for improvement to yield predictable results;
5. Obvious to try—choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and
6. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art.

There are guidance documents published by the USPTO to explain these and other analyses performed by the patent examiner. For example: Examination Guidelines for Determining Obviousness

under 35 U.S.C. 103 In View of the Supreme Court Decision in KSR International Co. v. Teleflex Inc.

2.5 The Possibilities of Means-Plus-Function Claims

The Supreme Court's decision in Alice Corp. made it more difficult for applicants to obtain patents for processes implemented in software, especially given the breadth with which the decision is being applied. After Alice, the rate of rejection of applications for software process applications has exceeded 80% in some of the USPTO's art units where it was previously below 40%. Means-plus-function claims, however, may provide some relief to applicants seeking to patent an algorithm couched as a process.

Title 35, § 112 (f) of the U.S. Code states:
an element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of . . . acts in support thereof, and such claim shall be construed to cover the corresponding . . . acts described in the specification and equivalents thereof.

These types of claims are known as "means-plus-function" claims because such claims usually begin with the words "a means for . . ." followed by a general description of the function of the invention. Some practitioners believe that means-plus-function claims are the best way to attempt to patent software-implemented processes in the wake of Alice Corp.

Computer programs are usually broken down into modules or subroutines characterized by a specific function. When software claims focus on the function of the modules, the protection of the patent is arguably broad enough to take into account the different methods of accomplishing the function thereby thwarting the ability of competing software programmers to accomplish the same function. Means-plus-function claims help avoid rejection due to abstract subject matter. The purpose behind means-plus-function construction is to clearly limit the scope of the claim to a particular physical implementation. A means-plus-function claim seeks to

encompass specific algorithms that transform¬ an otherwise general-purpose computer into a special-purpose computer programmed to perform the recited function.

The downside, however, is that means-plus-function claims provide limited protection given that the interpretation of what constitutes an "equivalent thereof" (from the patent statute) is not well-settled. It may be difficult, therefore, for an owner of a patent containing means-plus-function claims to demonstrate an equivalent function in an infringing process/algorithm. Furthermore, the function of a potentially infringing equivalent device must perform precisely the same function as the means-plus-function claim, leaving only insignificant differences in the way and result that the accused device functions, in order for infringement to be found.

2.6 Patent Costs and Filing

When applying for a patent, applications are subject to a payment of a basic fee as well as additional fees, including a search fee, an examination fee, and an issue fee. Excess fees are also due where applications include more than 20 claims. Patent application filing fees can be found at www.uspto.gov/learning-and-resources/fees-and-payment/uspto-fee-schedule#Patent%20Fees.

The USPTO fees are minimal in comparison to attorney fees to draft a patent. Although patents are attractive to investors and may protect a company's software, they are expensive to procure, and very expensive to defend if infringed upon. Patent prosecution expenses vary widely depending on the complexity of the application, but a range between $5,000 and $15,000 is representative for U.S. applications. Separate prosecution fees are payable for each country in which a patent is sought. According to the American Intellectual Property Law Association, where $1 million to $25 million is at risk, the cost of an average patent lawsuit is $1.6 million through the end of discovery and $2.8 million by the end of the trial.

This reality has led some software developers to rush to be the first to market, try and obtain users, and keep as much as possible about the code and architecture as a trade secret. It should be kept in mind that patents can be valuable outside of the context of infringement lawsuits, for example: (1) to demonstrate credible technology to potential investors, (2) to motivate competitors to design around the patent to avoid infringement, and (3) to stop infringement with the threat of an infringement lawsuit (defense of an infringement lawsuit is as expensive as the plaintiff's costs to pursue the case).

2.7 Time for Receiving a Patent

The time between filing and obtaining a patent can be three to five years, somewhat faster if a provisional patent application is not filed. Applicants can expect to hear from the USPTO about two years after applying due to a backlog of several thousand applications. This first communication is known as an "office action." The applicant will then respond to the action, followed by another response by the USPTO. The patent examiner and the applicant will typically communicate back and forth as patents are often rejected at first, and the applicant and attorney will negotiate with the examiner. By the end of this process, it will typically take about three years from the date of initial application to receive a patent.

2.8 Strengths of Protecting Software with a Patent

Companies seek patents for a number of reasons. Patent applications are a minimum requirement of some investors. Patents are valuable for defensive reasons if a company is accused of infringing another patent by a competitor or non-practicing entity. Patents can be a valuable negotiation tool should the company infringe another patent, enabling the company to offer a cross-licensing agreement with its patent for rights to the infringed patent.

2.9 Weaknesses of Protecting Software with a Patent

There are differing opinions about the value of software patents

for startup companies, even if one can be obtained. This is in part because software iterates quickly to address changing systems and challenges, and by the time a patent is granted, the software can be obsolete. It is not feasible for startups to seek protection for each iteration.

Another factor affecting the value of a software patent is that during patent prosecution, claims are often narrowed to such an extent they do not provide a value commensurate with the expense to procure them. Other concerns about seeking patent protection include publication of the inventive aspects of the process. In software, the architecture of the process becomes available to competitors regardless of whether the patent is granted or not, perhaps providing information that will help develop a competing product. In addition, patents that are granted can be challenged as invalid.

It may be difficult to determine whether a competing product infringes a patent because even if the competing software is examined, only the object code would be available, and it may be difficult for the patent owner to discover a case of infringement, thereby potentially nullifying the benefit of patent protection (or any kind of IP protection). An owner of a software process patent must still enforce the rights the patent affords, and to do so requires a way to detect potential infringement. Patents are sometimes infringed without the knowledge of the patent owner. When the patent owner is aware of the infringement, it takes money and time to enforce the patent and bring an infringement action.

3 Trade Secret Protection

Software is a good candidate for protection as a trade secret, because it is distributed to customers only in the form of object code, which cannot be read by people. Therefore, the source code, and possibly parts of the architecture (i.e. the functional process implemented by the source code), can be kept as a secret, access to which is limited to programmers, who can be required by contract to keep information about the architecture and the source code se-

cret. While patent protection requires the disclosure of the architecture, there is not a similar requirement for trade secret protection. Trade secret protection therefore can be applied to software code, and the architecture to the extent the process is not evident from using the software and interacting with the user interfaces.

However, trade secret protection does not eliminate the possibility that two developers could create very similar software to address the same problem or perform the same functions. The developer who is second to create the software has full rights to commercialize it so long as it was independently developed and not gained by improper means, which, in trade secret terminology, is misappropriation.

3.1 Elements of a Trade Secret

Trade secrets are defined by the Defend Trade Secrets Act of 2016, found at: 18 USC 1839 (3) as:
"all forms and types of financial, business, scientific, technical, economic, or engineering information, including patterns, plans, compilations, program devices, formulas, designs, prototypes, methods, techniques, processes, procedures, programs, or codes, whether tangible or intangible, and whether or how stored, compiled, or memorialized physically, electronically, graphically, photographically, or in writing if—
(A) the owner thereof has taken reasonable measures to keep such information secret; and
(B) the information derives independent economic value, actual or potential, from not being generally known to, and not being readily ascertainable through proper means by, another person who can obtain economic value from the disclosure or use of the information,"

A trade secret establishes its value by giving the owner of the information an economic advantage over competitors.

3.2 Reasonable Precautions to Maintain Secrecy

Once a company determines it has inventive software that represents economic value to the company, is not known to the public, and is not readily ascertainable by proper means such as reverse engineering, the company must take reasonable measures to maintain the secrecy of the source code and architecture if it opts to protect the software via trade secret. A company's mere intent that some information remain secret is not enough to fulfill the requirement. The company must take concrete, reasonable security measures to maintain the secret if it wishes to seek recourse in the courts for misappropriation.

Defining reasonable security measures for protection of a trade secret is challenging as courts determine what is reasonable on a case-by-case basis. However, the courts have outlined some basic requirements, including marking documents that describe the trade secret as confidential and limiting access to these documents with appropriate means, such as locked cabinets or rooms, safes, fences, or guards, depending on the circumstances. Everyone who is given access to the trade secret must sign a confidentiality agreement that identifies the information that must be kept secret. Additional measures include disclosing to the people who are given access to the trade secret only the portion of the trade secret necessary for them to do their work, distributing company phones or computers that must be returned to the company upon termination of employment to minimize the ability to copy trade secret documentation, employing encryption of software to make reverse engineering difficult, and utilizing software that enables self-destruction after detecting copying.

Courts have held that some disclosures, unprotected by nondisclosure agreements or other limits, did not void trade secret protection in cases where an implied confidential relationship existed between the company disclosing the trade secret and the recipient of the information. However, companies should not rely on implied confidential relationships to protect trade secrets; a written confidentiali-

ty agreement should be obtained.

3.3 Misappropriation

To establish a successful trade secret claim, a plaintiff company must show that the defendant misappropriated the trade secret in order to recover damages. There is no infringement remedy for trade secrets because trade secrets do not involve inherent property rights, such as the rights created by an issued patent. As explained above, trade secret rights do not prevent either independent invention of the subject matter of a trade secret, or reverse engineering of the trade secret by purchasing a product that embodies the trade secret and disassembling it to understand how it works. Misappropriation is established when the trade secret is acquired by improper means, such as stealing it or using it in violation of a confidentiality agreement.

The plaintiff must establish that the defendant had access to the trade secret (e.g., a rogue software developer), or gained access to the trade secret by deception (e.g., industrial espionage), and subsequently used the information without the permission of the owner. Use of a trade secret by another without permission is not automatically misappropriation. If a trade secret is discovered through reverse engineering, it is proper, and the original trade secret owner will not prevail in a claim against the second inventor because it was not obtained via misappropriation. If it is difficult to reverse engineer a software program, protecting it by means of trade secret is an attractive mechanism to keep a competitive advantage for a long period of time.

It may become necessary or financially beneficial for a company to disclose its trade secret to another. A company can do this and not waive its trade secret as long as reasonable precautions, such as nondisclosure agreements, are taken to protect the information. These types of disclosures allow outside parties to properly obtain trade secrets under a confidential relationship. Even though these trade secrets have been properly obtained, it is still possible for a

company to misappropriate the protected information if the company uses or discloses it in a way that was not agreed upon.

3.4 Advantages of Trade Secret Protection

Trade secrets are advantageous to software companies because protection lasts as long as the secret is maintained. Additionally, no disclosure of source code or process is necessary for the protection, unlike copyrights and patents, which both require disclosure. This makes trade secrets an attractive option for software developers, in spite of the fact that there is no protection against independent creation or reverse engineering. No registration or other interaction with a government agency, such as the USPTO, is necessary for trade secret protection of software.

3.5 Weaknesses of Trade Secret Protection

The disadvantages to protecting software as a trade secret include the fact that it does not prevent a competitor from inventing something similar or reverse engineering the program. If a trade secret is discovered through these legal means, there is no way to prevent its use by competitors. Thus, trade secrets are most valuable when it is unlikely or almost impossible that someone could reverse engineer the product or independently discover it on their own. Finally, once software is released and available to the public, other coders may independently create a similar program with their own code. Another drawback of using trade secret is that once it is discovered and published, or otherwise disclosed, it ceases to be a secret.

4 Copyright Protection

Copyright provides legal protection for original works of authorship fixed in a tangible medium of expression. Copyright law protects literary, dramatic, musical, and artistic works, such as poetry, novels, movies, songs, computer software, and architecture. Software source code is considered a written work, because it is

written in high-level programming language that can be read by people who understand the language. Registration of the copyright for source code also protects the object code version of the software, even though it is in the form of digital bytes that cannot be read by humans, because the object code is a direct translation of the source code, which is done in a standardized way by a compiler that presents the information contained in the source code. Copyright protects the right to copy, distribute, and create derivatives of the original, but it does not protect the underlying process executed by the software. Therefore, a software program that performs the same process carried out by different source code (i.e., not copied or derived from the original) does not infringe a software copyright, and software copyright protection is narrower than software patent protection.

Copyright comes into existence at the time of creation of the protectable work, and registration is not necessary to obtain the copyright in the protected work. However, to enforce the copyright, the work must be registered with the Copyright Office. Registration requires a disclosure of the work; for example, a copy of the book or the source code for the software. Copyright protection lasts for the life of the author plus 70 years after the author's death. If the work was created for a company pursuant to a "work for hire" agreement with an employee within the scope of their employment, the work receives copyright protection for either: (1) 95 years after the company registers the copyright protected work, or (2) 120 years after the company creates the work, whichever expires first.

The copyright holder must file a copy of software source code with the U.S. Copyright Office to register the copyright in the software. Registration is a simple process and enables the copyright holder to enforce the copyright by legal infringement action against a copyright infringer in federal court. Registration of a copyright requires completing an online application form, submitting the copy of source code, and paying a $35 to $55 filing fee (compared to the thousands of dollars required to obtain a patent). It is possible to redact certain portions of the source code to prevent revealing trade

secrets, but the extent of redaction is governed by specific rules, and the minimum amount of source code that is filed could reveal the trade secrets embodied in the software. The effectiveness of allowed redaction of source code for purposes of copyright registration must be evaluated on a case-by-case basis. Registering a software copyright will provide immediate protection if the work is copied.

4.1 Disadvantages to Copyright Protection for Software

Because a copyright only protects the source code itself, developers can create a different code that performs the same function, and they will not be infringing on the copyright. Copyright infringement may be difficult to detect and prove, especially with software, because it would have to be demonstrated that the exact code was copied. In terms of detection, in most cases an algorithm will not be widely disclosed for inspection and identifying an infringing algorithm could be difficult and highly speculative. Once infringement is detected, a plaintiff must still prove infringement in court. Furthermore, if source code is likely to become obsolete quickly, it may not make sense to register the copyright due to the short commercial lifecycle of the source code.

4.2 Copyright Infringement

To establish a cause of action in court for copyright infringement, a plaintiff must prove (1) ownership of a valid copyright, and (2) copying of original elements of the work.

Whether someone an accused defendant copied the plaintiff's copyrighted material is a factual question of whether the defendant actually used the copyrighted work to create his or her own work. A copyright violation is demonstrated through circumstantial evidence establishing: (1) access to the plaintiff's work and (2) probative similarities between the works. A defendant is able to rebut the evidence. For example, the defendant may be able to demonstrate he could not reasonably have had access to the plaintiff's work.

A correlation between two software programs/algorithms can be due to a number of things, including third-party source code, similar code generation tools, commonly used elements, a common author, or copying. Programmers may develop a program at one company then leave and independently develop a program at another company. This is perfectly legal and, if done correctly, does not constitute copyright infringement. Only copying that is unauthorized and substantial constitutes copyright infringement.