Santa Clara University School of Law

Santa Clara High Technology Law Journal

Volume 33 | Issue 3 Article 1

2-20-2017

Software as Text

John Shaeffer

SOFTWARE AS TEXT

John Shaeffer[†]

A computer program can be many things. As source code, it is unequivocally text and it is unpatentable written material. As text, it has syntactical structure, which allows a computer to respond to it, and meaning that it communicates to the programming community. A computer program is also an audio-visual work; we see it and hear it as we interact with it. In this way, protecting a computer program is like protecting music, which is often based on both the written score and the sounds we hear. But unlike modern music, which does not always depend on a written score, a computer program must always have source code before it can be heard.

Much has already been written about whether computer programs should be copyrightable, but little has been written about access. If a computer program is written in a compiled, rather than an interpretive language, the source code will be converted into assembly or directly into machine code, neither of which has much meaning to anyone but a computer. Access to the audio-visual expression of a computer program is different than access to its source code, with the former much more likely circumscribed to the programs unprotected ideas and functions. The audio-visual structure of a computer program is something very different than from the architecture and structure of its source code. For this reason, the infringement analysis in Lotus Development Corporation v. Borland[1]*—which was concerned with the identity in the visual structure of menus with no allegation of access to the source code used to depict the menus—has little in common with* Oracle v. Google[2]*—concerned with copying of standard tools used to write source code in the JAVA programming language.*

This article suggests that with a closer consideration of the practice of computer programming, intellectual property law

---

† John Shaeffer is currently a partner at Fox Rothschild LLP in Los Angeles, CA and is an adjunct professor at Santa Clara School of Law. The views expressed in this article are those of the author and do not necessarily reflect those of For Rothschild LLP or its clients.

1. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995).
2. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

governing the scope and extent of its protection can brought more in line with actual practice. With such an understanding, the law as applied to computer software can better serve its constitutional purpose of promoting the public good.

TABLE OF CONTENTS

INTRODUCTION

Source code is not functional. It is instructional. Source code describes one of several ways to obtain structured binary data that when sequenced to a processor in a particular order causes a computer

to perform particular functions. Programming a computer is creative in that the methods and styles used to execute the programmed functions say a lot about its creator.

In the same manner that a composer learns from studying the musical scores of her predecessors, a novelist learns from reading novels, and a filmmaker learns from watching film, programmers learn their craft from studying source code and other software architecture documents. This is why "[a]ll coding inevitably involves double coding. 'Good' code simultaneously specifies a mechanical process and talks about this mechanical process to a human reader."[3] This is a central premise underlying the Free and Open Source Software Movement (FOSS), a premise currently shared with the constitutional underpinning of this country's intellectual property laws that the public good is served by free and open access to the ideas and the expression of those ideas. "Free" here requires a precise definition: free as in "free speech" or "free trade," not "free beer."[4]

This article posits a disconnect between how programmers, and those who teach programming, view their craft, on the one hand, and the laws governing computer software and the academia supporting the governing legal structure, on the other. Much has already been written about the problems applying existing copyright and patent laws to computer programs as well as to the deficiencies in legal decisions making such efforts.[5] An area less explored, however, is whether the law fits with the practice it governs. Little consideration has been given to whether the law comports with what programmers are actually doing and the product they and their employers make and own.

The law curiously takes a myopic view of computer programs as being a utilitarian tool that allow computers to perform functions like word processing or reserving a seat on an airline. Programming is far more diverse than the law gives it credit for. Identifying the field of computer programming with the compiled copy of Microsoft Word is somewhat akin to characterizing all literature by referring to the design specifications for Boeing's latest jet—albeit that such design

3. Michael Mateas & Nick Montfort, *A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics*, *in* DIGITAL ARTS AND CULTURE: DIGITAL EXPERIENCE: DESIGN, AESTHETICS, PRACTICE (DAC 2005).

4. GEOFF COX & ALEX MCLEAN, SPEAKING CODE: CODING AS AESTHETIC AND POLITICAL EXPRESSION 77 (MIT Press 2012).

5. For example, some of the Federal Circuit continue to push to exclude computer programs from the breadth of section 101 of the Patent Act. Intellectual Ventures I LLC v. Symantec Corp., 838 F.3d 1307, 1328 (Fed. Cir. 2016) (Mayer, J., concurring) ("Declaring that software implemented on a generic computer falls outside of section 101 would provide much needed clarity and consistency in our approach to patent eligibility.").

specification would be more like the source code for Microsoft Word than the actual compiled program.

By way of example, the musical duo Autechre years ago dispensed with conventional instruments in favor the visual coding language MAX/MPS.[6] Their compositions today are distinct programs—referred to as patches—that result in computers performing functions that, with the help of digital to audio converters, generate complex sounds and rhythms. The spontaneity and creativity of an artist interacting with a guitar and keyboard is now left to a computer interacting with code in a spontaneous and not necessarily planned or expected way. Far from being an academic exercise, Autechre tours the world playing at large electronic dance music festivals and signed to a well-recognized music label.[7]

In this context, an Autechre patch functions like a musical symphonic score that instructs the members of the orchestra how and when to play a note. That the audience for both most likely cannot read either the score or the program does not detract from their expressive natures. No one would consider the score to Beethoven's Ninth Symphony to be utilitarian because it serves only to instruct musicians what to do in a particular time and order as directed by a conductor. Does this conclusion change when the sonic generator—here a computer program—replaces the musicians? This example is not isolated. Similar comparisons can be drawn between computer games and motion pictures, both of which typically start with a scripted story and through the functional manipulation of light sound and characters result in a visual narrative that few would consider utilitarian. This line becomes almost completely blurred when digital animation created on a computer can be ported over to either a digital film or an interactive game.

Law, and in particular our common law system, is evolutionary. The law builds on itself. In the 1976 Copyright Act, Congress took sides and adopted the recommendation of an expert panel to codify the copyright law's protection of computer software, a somewhat controversial decision at the time. Curiously, while this expert panel included some of the most noted authorities on copyright law, the panel did not include any experts in computer software.[8] Courts have

---

6. For a more in-depth discussion, see Joe Muggs, *Autechre: elseq et al*, RESIDENT ADVISOR (June 8, 2016), http://bit.do/Autechre. Autechre is merely one popular example of musicians using code as an instrument of sound. A listing of similar artists can be found at http://bit.do/alograve.

7. *Id.*

8. Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine Readable Form*, 1984 DUKE L.J. 663, 699 (1984) [hereinafter

struggled to apply the terms of this recommendation. Many challenge the decision that copyright law should govern computer programs, arguing instead that computer programs are by their nature functional. Therefore, they should be governed by patent law, if protected at all. During this same period, however, the Patent Bar has struggled with the extent to which patent law does and can reach computer programs.

This article offers a do-over.[9] Knowing what we know now, should computer programs be governed by copyright law? The answer will be yes. First, we will discuss software programming, what it is and how it is done, from the perspective of those who program and teach programming. True, the result of programming is software that causes a computer to be able to perform particular functions. But the process of programming is much more than simple machine code that when received by a processor in a particular order result in an output of binary data in a particular order. Again a simple example, software vulnerabilities are among the biggest threat to this nation's interconnected computer systems. The most common cause of a software vulnerability is an error in the coding of a commercially functioning program that can be exploited to give, among other things, an unintended person access to a system. Such vulnerabilities are typically fixed (closed) with a software patch consisting of different source code that when compiled will instruct the computer to perform the same function, but without the unintended exploitation, *i.e.* the machine code that can cause a computer to perform a particular function can be programmed innumerable ways. Achieving intended functionality at the machine level—whether it is to allow my word process to visually reflect my keystroke or to cause a particular note to be played at a particular time—can be achieved through a variety of instructions.

Software programming involves choices: choices of architecture, structure, paradigms and language. These choices are as frequently aesthetic—having no functional value—as utilitarian. With a more precise understanding of programming, the law can better achieve the constitutional goal of providing certain protections to certain

---

Samuelson, *CONTU Revisited*] ("There were no computer scientists, no software or hardware industry representatives, nor any users of complex software systems on the Commission. Because Congress had decided that the composition of the Commission should reflect the interests of groups concerned about the computerization of hard-copy works and about photocopying, this is not surprising.").

9. *Id.* at 754 ("Lawyers, judges, and legislators often find it difficult to step back and take a long, hard look at the law to determine whether it 'works' as it was intended to work. Many persons closely involved with the software industry believe that the existing means of legal protection for computer programs in machine-readable form do not work satisfactorily.").

intellectual property in a manner that furthers the public good. Simply, like anything, there is no "one size fits all." Case law concerning copyright liability arising from mimicking functionality based on observations of a compiled program in use, has little if any application to copying functionality from a review of source code or software design documents.

Next, copyright law at a basic level will be considered. Existing jurisprudence and academic research in the area of copyright protection for computer software tends to focus on copyrightability and substantial similarity, while ignoring other elements necessary for liability. Decision after decision struggle with whether what was purportedly copied was copyrightable expression as opposed to an unprotected idea, function, structure, or process. Unlike other categories protected by copyright, a large portion of publicly available software programs intentionally block access to all but the most occult elements of what is literary about a program. In this way commercial software programs are unique. As part of the copyright bargain, the computer program discloses its unprotected ideas, functions and process, but intentionally hides its written expression—how it teaches the ideas, processes and functions disclosed. While copying the entirety of a compiled program necessarily takes both what the program teaches and how it teaches it—and is universally considered copyright infringement—it is another thing to discern how a program teaches a particular idea, function or process simply by reviewing its assembled code or observing its operation.

Little can be learned about how its creator(s) expressed the instruction that achieved the requisite function by reviewing either the assembled code or the binary machine code. This means that literally copying anything short of the complete executable program will be of little value and thus unlikely. Few, if any cases, accuse a defendant of copying portions of either the assembled code or the more basic machine code in order to achieve functionality. Instead, in such cases it is argued that observing functionality is a surrogate for access to the expression of that functionality—how the functionality is taught in the source code. By analogy, this is like someone telling me the general plot of Star Wars from which I write my own Parsifal-like space movie. Arguably, enough could be conveyed in the recitation to trigger liability, but there is a gap in access between the idea (or function) expressed and how the idea (or function) was actually expressed

Few would disagree that I would commit copyright infringement if I copied my neighbor's DVD of Call of Duty in order to play the

game on my computer, or that this type of copying is any different than coping an .mp3 of the latest Beyoncé album or using a torrent to download the latest Star Trek movie. Difficulty and disputes arise when copying is not exact and the allegation is whether the new work was derived from the protected work in a manner that violates copyright law, *i.e.*, the new work is a protected derivative work. Consistent with the most basic principles of copyright law, if what is taken is so far removed from protectable expression to be only the ideas, facts, functions or processes of the underlying work, rather than how these elements were express, then there is no infringement. Drawing this line, however, is what is difficult.

The copyright law gives the holder of a copyright the right to prevent others from copying their work. When copying is given, infringement has occurred, but may be defensible as a fair use. For example, if what was copied was simply code allowing an application to work with an operating system—interoperability—or simply code that helps programmers to write in a given language—as was the case in *Oracle v. Google*—the copying may actually further the public good rather than result merely in a new work that substitutes for the protected work. Allowing a defendant to prove her copying fair—for example to achieve interoperability—will be shown to be an easier approach to address thorny issues of permissible copying than to argue in such instances particular code is no copyrightable because, for example, its expression has merged with what should be a publicly available function.

The Copyright Act does not include a *de minimis* exception. While a use may be fair, an initial prima facie case of infringement cannot be defeated by arguing how little was copied. The substantial similarity test—which has been used by some to support a *de minimis* exception—governs only when there is no direct evidence of copying, allowing copying to be presumed when two works are so similar that independent creation is unlikely. "Substantial similarity" is not a separate test. So since Congress has decided to protect computer programs under copyright law, when actual copying is acknowledged infringement has occurred, but the use may be fair, as in the case of copying necessary for needed interoperability.

The article will also discuss patent law for the sake of comparison. Like copyright law, patent law does not protect ideas, processes or functions, but instead protects them as used. For this reason, patent law does not protect source code, which consists merely of written instructions. Patent law can protect the ultimate function that the compiled code allows the computer to practice, so

long as the code enhances the functionality of the computer in a unique way. There is, however, a meaningful distinction between the function that the source code enables the computer to perform and the instructions, structure and architecture of the source code itself that once compiled allows that function to occur. Rather than there being any risk of overlap between patent and copyright law with respect to software programming, there is a gap between the two because, despite the simple language of Section 101 of the Patent Act, not all ideas, functions or processes are patentable and copyright law does not protect ideas, functions or processes.

Finally, the article proposes that copyright law should govern most of what falls within the practice of software programming. Arguable, the public at large has suffered due to the ambiguity surrounding copyright law's role. Unsure of the legal protection surrounding source code in particular, technology companies dependent upon their developed programs to survive and prosper lock their source code from public view, thus denying the public access to the ideas expressed in that code. Placing computer programs in the unique position of being able profit from their ideas and function without teaching how they achieve the functionality—the idea or function is disclosed while only disclosing a minimal amount of the expression of that function. In direct response, FOSS use copyright law for the express purpose of avoiding this practice, requiring under certain licenses that the use of protected code be conditioned on the user similarly making publicly available their source code.

Aware that the existing jurisprudence has made application of copyright law to computer program difficult, the article suggests looking to the rest of copyright law jurisprudence to assess the breadth of relevant protection. For example, the earliest days of this country's copyright law dispels any notion that copyright law does not protect works that have utility. As initially enacted, the Copyright Act protected the expressive elements of charts and maps irrespective of the fact these were unquestionably works of utility. Similarly, questions about whether copyright law should reach the structure and architecture of a software program's design is readily answered by how works of literature and film are assessed for infringement, which typically turns on structural considerations rather than proof of literal copying.

Concerns about the overlap between expression and function that purportedly plagues the protection of software programs can be addressed by looking to the scope of protection afforded works of science and non-fiction. Copyright law does not protect the facts,

formulas, processes or compounds discussed in non-fiction and academic literature, but only how the same is expressed. Stated simply, copyright law does not protect *what* is taught, but instead, protects *how* it is taught. Anyone involved with pedagogy knows there exist enumerable texts teaching the same subject matter, but teachers, often subjectively, prefer one text over another. Those familiar with historiography know that the telling of history is not simply collecting facts in chronological order. Facts relevant to any historical event are enumerable, but the same history can be told in multiple ways with multiple conclusions depending on the politics of the speaker. The creativity that drives success in these other areas drives success in programming. There is simply not one way to instruct a computer on how to perform any given function, and programmers learn from each other by studying approaches for structuring and designing source code that can cause a computer to perform particular functionality.

As most programmers will readily acknowledge, the success of a computer program is not driven by a code's elegance or efficiency, but ultimately by its consumer's preference. The most efficient and functional program is of little or no value if its intended consumers do not use it, a preference driven more by the aesthetics of taste than anything else. So while computer programs are unquestionably about function and utility, what differentiates programs is about much more. Most people today have a preference for Apple, Microsoft, android or LINUX, and many are passionate about their positions. Yet, when pressed, these preferences are more about taste than objective functional differences, little different than a preference for Taylor Swift over Mozart, or fish over chicken.

Shoring up copyright's governance of computer programs, however, is not without problems. In particular, as an artifact of the laws development, creators can wait for years or decades before publicly disclosing their rights in a work and asserting such rights. The collaborative nature of many popular software programs—incorporating existing publicly available code that preforms specific rudimentary tasks rather than recreating the wheel—makes this ability to lie and wait particularly problematic. This ability to disrupt the availability of programs is a leading reason why many advocate against copyright law governing computer programs. This anomaly, however, can be addressed by amending the Copyright Act to include a prompt "register or lose it" requirement as well as an explanatory marking requirement.

I. WHAT IS COMPUTER PROGRAMMING?

A. Language As Instruction

All language by its nature is instructional. In this way, all language is functional. When we say, "Can I have a cup of coffee?" we expect a functional response—we will receive a cup of coffee. Even purely aesthetic speech is functional. Shakespeare had ideas of human nature that he wanted to convey to an audience. He utilized the rules of the English language, its syntax and grammar, for a semantical end. This end is no different than when we are attending a cocktail party and recounting the day's events. We intend to convey meaning—we want to be understood. How our speech is understood—its function—however, many not always be apparent. There is no such thing as neutral speech.[10] Whether intended or not all speech invokes action by the recipient, even if the listener simply concludes "that was a dull story" (which was probably not the function the speaker intended to invoke). The process of programming makes the functional nature of language concrete—code both says what it is going to do and does it.[11]

Language is like money in that it is nothing, but it affects everything.[12] Language is unique in that it is the only concept that cannot be described without using itself. Language, however, is inherently imprecise. It can never accurately convey meaning. Language is a mediating tool between our thought and the recipient of the words conveyed.[13] While each language has its rules aimed at achieving semantical precision, there is no objectively optimal way to convey anything. This is one reason why speaking in the presence of someone is less likely to be misinterpreted—convening meaning not intended—than more remote forms of communication such as over the phone or by email.[14] Visual and auditory clues accompanying language add to its precision in ways that an emoji cannot convey.

---

10. LUCE IRIGARAY, TO SPEAK IS NEVER NEUTRAL (Gail Schwab trans., Routledge 2002) (1985) (demonstrating that the language use in peer review scientific article is political and not neutral).

11. COX & MCLEAN, *supra* note 4, at xii.

12. *Id.* at x.

13. RICHARD RORTY, PHILOSOPHY AND THE MIRROR OF NATURE (Princeton Thirtieth-Anniversary ed. 2009) (1979); Chris van Rompaey, *Language and Meaning in the* Ethics*. Or, Why Bother with Spinoza's Latin?*, 24 PARRHESIA 336–66 (2015) (demonstrating how meaning is lost in translating Spinoza from Latin to English).

14. Curiously programmed listening devices like Apple's Siri or Amazon's Alexa are experimenting with visual mechanisms that will allow the computer to look at the speaker to better understand the request made. Will Knight, *Chatbots with Social Skills Will Convince You to Buy Something*, MIT TECH. REV. (Oct. 26, 2016), http://bit.do/KnightChatbots.

The existence of our legal profession is a testament to the imprecise functional nature of language. Legislatures pass laws for the functional purpose of defining rules for a particular situation. By their nature, however, laws written generally will necessarily effect specific situations arbitrarily.[15] Every business days reviewing courts throughout the United States, including the United States Supreme Court when in session, consider written and oral position of opposing sides each of whom believe their use of language will achieve for them a function, a decision in their favor. The language presented has a function, to persuade towards a position. Neither the parties' briefs nor the decision by any court are objectively optimal, but rather are always political. They advocate for an end, but what is conveyed can never be objectively optimal, a point demonstrated when someone invariably says, "they got it wrong."

Language can also be an intermediary to function. We know that a written play or a motion picture script are instructions for a performance; *i.e.*, we can read a play or a script and envision the function that that language enables. The language is necessary to achieve the function and in this way a script is utilitarian. In this way, a script is like the blue print for a building—each provides instruction on how to achieve a desired product—either a motion picture or a home.

The aesthetic pleasure we derive from a film or a novel is not limited to the story told. In fact, it is much more about how the story is told, its mood, pacing, structure. Many would scoff comparing *Romeo and Juliet* or *West Side Story* to the *Twilight* series, but are they not telling the same story, built on the same idea, just expressed in different ways? Every year thousands travel to the Bayreuth Festival and watch "new production" of one or more of the thirteen operas written by Richard Wagner and critics will debate endlessly the merits, or lack thereof, of the latest staging. Similarly, historians do not consider Edward Gibbon's *The Decline and Fall of the Roman Empire* to be one of the greatest works of non-fiction because of its factual accuracy or conciseness.

Our ability to understand language says nothing about whether a writing that enables a function is simply utilitarian. The fact that the language of physics and math rely on numbers and symbols for expression do not render the symbols and numbers merely utilitarian objects to achieve a function. A better example is a symphonic score. Many of us could identify Mozart's Requiem by hearing an excerpt, but few of us could identify the work if given a page from its score.

15. H. L. A. HART, THE CONCEPT OF LAW (Oxford Univ. Press 2d ed. 1994) (1961).

Most would agree that the value of the Mozart's score is that it allows a conductor and trained musicians to play the piece, it functions as an intermediary between Mozart's idea and the musicians, but no one would say that our inability to similarly understand the score renders the score a simply utility. Does this analysis change when today a computer can convert this score into a MIDI file that allows a computer to cause a listener to hear Mozart's Requiem.[16] In both instances, the instructions are hidden from the audience, but their function is apparent, we hear music.

Language in all its forms, whether English, Chinese or musical notations, or a mathematical proof consists of rules governing symbols for the purpose conveying meaning.[17] "The essential property of language is that it provides the means for expressing indefinitely many thoughts and for reacting appropriately in an indefinite range of new situations."[18] Syntax "refer[s] to the rules of a language for the grouping of words into a larger units, *i.e.* sentences."[19] From the study of syntax of a language one can construct a "grammar" or the rules and principals applied to efficiently and effectively use the language to communicate.[20] "[G]rammar is autonomous and independent of meaning."[21] The study of "generative grammar" "attempts to characterize in the most neutral of terms the knowledge of the language that provides the basis for actual use of language by a speaker-hearer."[22] Semantics refers to the meaning of the words conveyed.[23]

---

16. *See PhotoScore MIDI*, SIBELIUS (Nov. 3, 2016), http://bit.do/SibeliusPhotoScore.

17. Noam Chomsky, *On Certain Formal Properties of Grammars*, 2 INFO. & CONTROL 137, 137 (1959) [hereinafter Chomsky, *Formal Properties of Grammars*] ("A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols."); NOAM CHOMSKY, ASPECTS OF THE THEORY OF SYNTAX v (MIT Press 1965) [hereinafter CHOMSKY, THEORY OF SYNTAX] ("The idea that a language is based on a system of rules determining the interpretation of its infinitely many sentences is by no means novel.").

18. CHOMSKY, THEORY OF SYNTAX, *supra* note 17, at 6.

19. Raymond Hickey, Syntax, 1 (Nov. 2016) (unpublished manuscript), http://bit.do/HickeySyntax; NOAM CHOMSKY, SYNTACTIC STRUCTURES 11 (Mouton de Gruyter 2d ed. 2002) (1957) [hereinafter, CHOMSKY, SYNTACTIC STRUCTURES] ("Syntax is the study of the principles and processes by which sentences are constructed in particular languages. Syntactic investigation of a given language has as its goal the construction of a grammar that can be viewed as a device of some sort for producing the sentences of the language under analysis.").

20. PAWEŁ LULA ET AL., PROGRAMMING METHODS AND TOOLS (Jan Madej ed., Cracow Univ. of Econ. 2014); Chomsky, *Formal Properties of Grammars*, *supra* note 17, at 137 ("A grammar can be regarded as a device that enumerates the sentences of a language.").

21. CHOMSKY, SYNTACTIC STRUCTURES, *supra* note 19, at 17.

22. CHOMSKY, THEORY OF SYNTAX, *supra* note 17, at 9.

23. NOAM CHOMSKY, LANGUAGE AND MIND 102 (Cambridge Univ. Press 3d ed. 2006) (2005) [hereinafter CHOMSKY, LANGUAGE AND MIND] ("At the crudest level of description, we

Linguists are those who study the grammar, syntax and semantics of language. Traditionally, they have developed theories of grammar from the study of existing languages. The work of linguists with natural languages, and in particular the work of Noam Chomsky, have profoundly influence the development and understanding of programming or artificial languages.[24] Rather than identifying the grammar of an existing language, programmers use the theories and rules identified by linguists as necessary for a language to build artificial languages.[25] In this context, building a language that can both speak to, and be understood by, the community of programmers and simultaneously instruct a computer how to act—*i.e.* a useful artificial language.[26] In this context, programming language make use of Chomsky's notion of transformational grammars. The idea that each sentence in a language has two structures: a deep structure—that determines the semantic interpretation of a sentence; and a surface structure—the syntactical representation,[27] stated very simply, the distinction between what is meant and how it is said. Linguists in their study presume "an ideal speaker-listener, in a completely homogeneous speech community, who knows its language perfectly and is unaffected by such grammatically irrelevant conditions as memory limitations . . . and errors . . . in applying his knowledge of

---

may say that a language associates sound and meaning in a particular way; to have command of a language is to be able, in principle, to understand what is said and to produce a signal with an intended semantic interpretation.").

24. Fotis Georgatos, How Applicable Is Python as First Computer Language for Teaching Programming in a Pre-University Educational Environment, from a Teacher's Point of View? (June 2002) (unpublished Master's thesis, Universiteit van Amsterdam) ("Noam Chomsky's Generative Theory of Grammars has, probably, influenced Computer Science more than the field of Natural Languages itself.").

25. Jukka Paakki, *Attribute Grammar Paradigms—A High-Level Methodology in Language Implementation*, 27 ACM COMPUTING SURVEYS 196, 199 (1995) ("Designing a programming language is an intellectual challenge of considerable complexity (for an overview, see Wasserman [1980]). One of the most important considerations is the description of the language. The description must be illustrative enough for a language user, and precise enough for a language implementor.").

26. PAWEŁ LULA ET AL., *supra* note 20, at 35 ("The construction of these artificial languages is currently accompanied by the construction of their grammars, which are strongly formalized from the beginning. Thanks to this, the translation process, meaning the automatic translation of a program designed to be convenient for humans (that is, the one expressed in an algorithmic language which is clear and understandable to the programmer) to the so-called internal code (binary strings controlling the operation of a machine, mainly microprocessors) can be conducted quickly and efficiently.").

27. CHOMSKY, THEORY OF SYNTAX, *supra* note 17, at 16 ("Consequently, the syntactic component of a grammar must specify, for each sentence, a deep structure that determines its semantic interpretation and a surface structure- that determines its phonetic interpretation. The first of these is interpreted by the semantic component; the second, by the phonological component.").

the language in actual performance."[28] With a computer the speaker has the linguist's assumed listener who will respond literally to each instruction as syntactically presented, meaning precision in a language and speaker is not merely assumed but required.[29]

B. What Is a Computer Program?

It is significant that the Copyright Act defines "a 'computer program' [a]s a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[30] The Legislature adopted the language of literature rather than words of mechanization, such as methods, process, rules or algorithms. As such, computer programs are protected under the Copyright Act from the perspective of their author as literary works. "Literary works,', according to the Act are "works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied."[31]

Most of us do not experience computer programs from the perspective of an author. At least since the advent of icon driving visual programs, our experience as end users of software is not a literary experience, but an audio visual experience. We do not read a program, but physically point and click on icons or identifiers to which we ascribe meaning. The Copyright Act separately protects audiovisual works, defining them as "works that consist of a series of related images which are intrinsically intended to be shown by the use of machines, or devices such as projectors, viewers, or electronic equipment, together with accompanying sounds, if any, regardless of the nature of the material objects, such as films or tapes, in which the works are embodied."[32] Courts have not hesitated to recognize that the Copyright Act protects the expressive audio visual components of a computer program, such as in a video game, but this protection does

28. *Id.* at 3.

29. LULA ET AL., *supra* note 20, at 35 ("Correctness of the various language structures is dealt with by syntax. It turns out that it is syntax that is most important in computer science, because the entire practice of using artificial languages (the so-called algorithmic languages), which are the basis of modern methods of programming, is based on it.").

30. 17 U.S.C. § 101. This definition is remarkably similar to definitions commonly found in programming tests. LULA ET AL., *supra* note 20, at 126 ("A computer program consists of a sequence of statements executed by the computer.").

31. 17 U.S.C. § 101.

32. *Id.*

not substitute or replace the separate protection of the computer programs as literary works.[33]

Similarly, some scholars attempt to put computer programs under the rubric of copyright protection for "pictorial, graphic and sculptural works" which include "two dimensional . . . works of fine, graphic and applied arts, . . . maps, globes, charts, diagrams, models and technical drawings, including architectural plans."[34] Following then existing jurisprudence, Congress expressly limited copyright protection for such works "only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspect of the article."[35] If the pictorial representations exhibited by a software program cannot be separated from their utilitarian functionality—as was the case in *Lotus v. Borland*[36]—the depictions, are not copyrightable. This reasoning,

---

33. Micro Star v. Formgen Inc., 154 F.3d 1107, 1112 (9th Cir. 1998) ("Micro Star further argues that the MAP files are not derivative works because they do not, in fact, incorporate any of D/N-3D's protected expression. In particular, Micro Star makes much of the fact that the N/I MAP files reference the source art library, but do not actually contain any art files themselves. Therefore, it claims, nothing of D/N-3D's is reproduced in the MAP files. In making this argument, Micro Star misconstrues the protected work. The work that Micro Star infringes is the D/N-3D story itself—a beefy commando type named Duke who wanders around post-Apocalypse Los Angeles, shooting Pig Cops with a gun, lobbing hand grenades, searching for medkits and steroids, using a jetpack to leap over obstacles, blowing up gas tanks, avoiding radioactive slime. A copyright owner holds the right to create sequels."); Tetris Holding, LLC v. Xio Interactive, Inc., 863 F. Supp. 2d 394, 408-11 (D.N.J. 2012) (although undisputed that no code was copied, the court found the audio visual displays of the games sufficiently similar to infer copying and infringement); *compare* DaVinci Editrice, SRL v. Ziko Games, LLC, No. H-13-3415 (S.D. Tex. 2016) (finding audio visual elements copied in games were unprotected rules and ideas rather than creative expression).

34. 17 U.S.C. § 101; Wendy J. Gordon, *How Oracle Erred: Functionality, Useful Articles, and the Future of Computer Copyright* 41 (2016), http://bit.do/GordonOracleErred. A shortened version "How Oracle Erred" will be published in the anthology, COPYRIGHT LAW IN AN AGE OF LIMITATIONS AND EXCEPTIONS (RUTH OKEJIDI ed., forthcoming 2016) ("While neither of these facts about the use of computer programs is sufficient to resolve the Oracle case, they remind us computer code may be a 'useful article' and that like all useful articles, considerations of patent deference can and should play a strong role.").

35. 17 U.S.C. § 101; Mazer v. Stein, 347 U.S. 201, 217 (1954) (holding statuette base of a lamp copyrightable even though the entire product functions as a lamp); Am. Dental Ass'n v. Delta Dental Plans Ass'n, 126 F.3d 977, 980 (7th Cir. 1997) ("A lamp may be entirely original, but if the novel elements are also functional the lamp cannot be copyrighted. This is not a line between intellectual property and the public domain; it is a line among bodies of intellectual-property law.").

36. *Lotus*, 49 F.3d at 817 ("Computer programs, unlike VCRs, are copyrightable as 'literary works.' 17 U.S.C. § 102(a). Accordingly, one might argue, the 'buttons' used to operate a computer program are not like the buttons used to operate a VCR, for they are not subject to a useful-article exception. The response, of course, is that the arrangement of buttons on a VCR would not be copyrightable even without a useful-article exception, because the buttons are an uncopyrightable 'method of operation.' Similarly, the 'buttons' of a computer program are also

however, says nothing about whether the protectable expression computer program itself, as defined in the Copyright Act, is copyrightable.

The "statement or instructions" to which the Act's definition of a computer program refers is commonly ascribed to the "code" that makes up the program, but should not be limited to the code itself. Most of us at best have a rudimentary understanding of what "code" does and have even less of an understanding of what it is. By way of analog, we can identify Chinese characters and know that the characters convey meaning, but most of us do not know what they mean. We have no understanding of the syntax or grammar of Chinese language so it is semantically meaningless to us. Simply because we cannot comprehend the Chinese language or it structure, however, does not render the language or works created with it simply functional and utilitarian and deserving of only minimal copyright protections.

The analogy between the Chinese language and computer programming is apt in this context. John R. Searle, the noted philosopher, proposed the following thought experiment. A person who does not understand Chinese is given questions written in Chinese characters. "He transfers the Chinese Characters into other Chinese characters solely through the completely formal, step-by-step guidance of an English rule book. Thanks to the precise transformation rules, the results are flawless Chinese replies to Chinese questions."[37] While Professor Searle experiment was directed to whether a computer has intelligence—it responds to syntax and grammar without grasping the underlying semantics[38]—the experiment is relevant to our understanding to the semantics of code. Unable to grasp the meaning of either the written source code or the operational machine code, our understanding of a computer program is limited to the functions it causes the black box we know as a computer to perform.

---

an uncopyrightable 'method of operation.'").

37. FLORIAN CRAMER, WORDS MADE FLESH: CODE, CULTURE, IMAGINATION 107 (2005).

38. COX, *supra* note 4, at 31 ("Searle's position is based on the linguistic distinction between syntax and semantics as applied to the digital computer or Turing machine as a 'symbol-manipulating device,' where the units have no meaning in themselves (a position that follows from semiotics)."); CRAMER, *supra* note 37, at 67-68 ("In 1948, Claude Shannon, a telecommunication engineer at the AT&T Bell Labs, coined a concept of information that did away with all semantics. It made information a technically quantifiable, measurable entity for determining (a) the transmission capacity of a channel and (b) the technical redundancy of data.").

Since most of us do not believe in magic, unfamiliar syntax and grammar without a grasp of semantics is easily categorized as purely utilitarian—I can perform the desired function by simply hitting the key on a keyboard in order to see words appear on the screen. Our lack of semantical understanding of the instructions does not alone render the instructions purely utilitarian. If the travel guide I am using to navigate my way through Berlin happens to be all in German, but I can match street names with the names in my guide even though I do not speak German, does not render the guide solely utilitarian. Similarly, the fact the music I am now listening to is simply binary code that my computer can read and translate into the sound I am now hearing does not mean that the song I am now listening to is simply utilitarian. Stated starkly, no one would accept as a defense to copyright infringement by a torrent that proves access to copyrighted material the argument that all it does is provide access to binary code.

Much the jurisprudence concerning computer programs focuses on the end product, the function the program performs for the end user rather than from the author's literary process resulting in that functionality. In the same way that there is no objectively superior way to teach someone math, English, physics, or software engineering there is no objectively superior way to instruct a computer to perform a particular task, whether that task is a new online reservation system for an airline, a video game to compete with Call of Duty, or code to oscillate light and sound based on movement in a room for an installation at the Pompidou in Paris. As will be discussed below, copyright is not concerned at all with what any of these teach. Instead, copyright will protect original way each is taught.

C. How We Interact with Computer Programs

1. How Consumers Interact with Computer Programs

Computer programs are everywhere. "Wherever there is a microprocessor, there is software."[39] Microprocessors are not confined to what we would perceive as computer or its progeny—cell phones and tablets—but are now present in our cars, traffic lights and coffee makers. Some would say that the highly trained engineers and machinists who historically designed, developed, and manufactured

39. ERIK PIÑEIRO, THE AESTHETICS OF CODE: ON EXCELLENCE IN INSTRUMENTAL ACTION 28 (2003). The recent denial of service attack that crippled a meaningful portion of the internet is a testament to this now that it has been discovered that the botnet consisted mostly of web-enhanced devices like cameras and refrigerators. Such devices are an easy target because users rarely change the factory default passwords for these devices.

machines for particular purposes have now been replaced by software engineers who design and develop code that allow a generic machine—or more specifically a generic microprocessor—to perform specific tasks or improve upon the products of the past by making them "smart." This analogy of code with "gears, wires, and screws" allows us to see code as simply functional industrial design, something not considered within the realm of copyright protection.[40] A programs, however, is unlike any other form of industrial design because "it is a machine built of text."[41]

Importantly, the Supreme Court has expressly rejected this notion of code as a "component" like a gear or screw. Specifically, the Supreme Court in *Microsoft Corp. v. AT&T Corp.* held that "code" is not a "component" within the meaning of section 271(f) of the Patent Act, which provides that infringement does occur even if a "product is made or sold in another country … when one 'supplies … from the United States,' for 'combination' abroad, a patented invention's 'components.'"[42] Instead, the Supreme Court considers "software code [to be] an idea without physical embodiment, and as such, it does not match § 271(f)'s categorization: 'components' amenable to 'combination.'"[43] Software, according to the Court, is a "'set of instructions . . . that directs a computer to perform specified functions or operations.'"[44] The Court considers code to be more akin to "[a] blueprint [that] may contain precise instructions for the construction and combination of the components of a patented device, but it is not itself a combinable component of that device."[45]

While we rely on media to house this text, the text itself is amorphous. With most industrial design, the finished product discloses not only how it works, but the methods employed to achieve that functionality. Commercial software for the most part is different. While we are told how to make the software work, the commercialization of the product does not similarly require full

40. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COL. L. REV. 2308, 2321 (1994) [hereinafter Samuelson et al., *Manifesto Concerning Legal Protection*].

41. Dan L. Burk, *Patenting Speech*, 79 TEX. L. REV. 99, 119 (2000); Dan L. Burk, *Method and Madness in Copyright*, 3 UTAH L. REV. 587, 614 (2007) ("For our purposes here this is significant because construction of a device by writing text is equivalent to constructing the device by soldering metal or molding polymers: both will execute processes to produce a determined, monovalent outcome.").

42. Microsoft Corp. v. AT&T Corp., 550 U.S. 437, 441 (2007).

43. *Id.* at 449.

44. *Id.* at 447 (quoting Fantasy Sports Properties, Inc. v. Sportsline.com, Inc., 287 F.3d 1108, 1118 (Fed. Cir. 2002)).

45. *Id.* at 450.

disclosure the expressive text employed to achieve that functionality. While our hard drives include program folders housing files needed for the software to operate, opening those files and viewing snippets of the assembled language tells us little about the textual instructions used to achieve this functionality.. With the appropriate software we can also view and access the object code—the binary string of on and off positions that causes a processor to perform a particular task—but the ability decompile from this string the instruction that when compiled achieved this functionality is fraught with difficulties.[46]

Unlike almost any other form of creative or industrial endeavor, one cannot copy snippets of object code from a complied program or grab files from the assembled code and insert them into a new program and expect to achieve any sort of functionality. Copying a compiled program, therefore, is an all-or-nothing affair. Many of us can attest to this when we have been unable to launch a program consisting of millions of lines of code simply because we mistakenly deleted a single .dll file from a program. Copying the entire executable, which allows the copier to utilize the program is universally considered copyright infringement and is clearly something Congress intended copyright law to prevent.[47] Actually copying something less than the entire compiled code, while still copying within the meaning of the Copyright Act, has little practical utility and rarely the subject of any copyright case. For this reason, programming is compared to "building [with] toothpicks carefully glued together . . . if just one toothpick is out of place the whole thing comes crumbling down."[48]

---

46. Atari Games Corp. v. Nintendo of America Inc., 975 F.2d 832, 843-44 (Fed. Cir. 1992) (Atari, unable to accurately determine the lock for game cartridges improperly obtained a copy of the program's source code to correct the errors it made attempting to decompile the object code).

47. Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 193 U. OF DAYTON L. REV. 975, 975 (1994) ("The debate is not over copying of code for resale or for the purpose of creating a usable second program to accomplish the function intended by its author. Copying of this type constitutes copyright infringement, and copyright laws work well as a legal prohibition of that kind of piracy."); Gordon, *supra* note 34, at 41-42 ("Copying an entire copyrighted computer program will of necessity make use of both expressive and nonexpressive aspects, and using the copy—even using it to run a machine—could therefore infringe."); Jonathan Ambrose, *Oracle America Inc. v. Google, Inc.: The Only Nonliteral Aspect of JAVA API's Protected Under copyright law are the Ones Nobody Wants to Copy*, 14 N.C. J.L. & TECH. ON. 1, 22 (2012) ("Two reasons exist for copyright protection as a method for protecting computer programs from infringement. First, the ease and utility with which computer code can be copied necessitate that direct, mechanical copying be protected by copyright law. Compared to mechanical inventions, computer programs are quick, easy, and inexpensive to copy.").

48. PIÑEIRO, *supra* note 39, at 216.

Our perception of a computer program is derived from how we interact with programs. "We are used to consider[ing] software as a tool, and we usually come in contact with it under the form of an application. For instance, we see the user-interface and through it we interact with an invisible system."[49] Our experience is limited to utility and functionality, with the instructions achieving that functionality hidden from view.

We can observe a program's function and ask another programmer to write for us a program that performs the same observed function. In this context, however, what is copied is the unprotected function, process or idea because we never saw the text that expressed this function process or idea.[50] Even if somehow the programmer's source code to mimic the observed functionality turns out to be identical with the hidden source code, there would be no infringement of the source code because the programmer could not have copied the source code.[51] This is no different from me asking an artist to recreate an unphotographed tapestry of Buddha sitting under a lotus tree found in a remote Tibetan temple based only on my recollection of what I saw. At best, the artist could copy the idea conveyed, but without access to the creative work itself, the artist could not copy any protected expression of the idea. Simply, you cannot copy the text, or create a new work derived from the text or its structure, without meaningful access to the text itself.

Exposed to only the utility and functionality of a program, and without an understanding of process of programming, we expect that

---

49. *Id.* at 42.

50. This is an explanation for the holding in *Lotus*, 49 F.3d at 810, in which the defendant unquestionably copied the visual depiction of the command hierarchy of the spreadsheet program Lotus 1-2-3 for its competing software program Quatro Pro. *See* Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement* 20 (2015), http://bit.do/SamuelsonFunctionality [hereinafter Samuelson, *Functionality and Expression*] ("Lotus v. Borland is the best known of the cases addressing whether a command structure of a computer program user interface (UI) is protectable by copyright law."). The visual depictions of the command hierarchy at issue in *Lotus* are fundamentally different than the hierarchy and structure of the source code that achieves this functionality. There is no dispute that in *Lotus* the defendant did not have access to the underlying source code for Lotus 1-2-3 and had not attempted to gain access through a review of the program object code or its assembled code. Instead, what was copied was how the program visually depicted its hierarch, which is something different than the instructions the program follows to achieve this functionality, which is the definition of a computer program under the Copyright Act.

51. *Mazer*, *supra* note 35, at 217-18 ("The distinction is illustrated in *Fred Fisher, Inc. v. Dillingham*, 298 F. 145, 151, [(SDNY1924) (L. Hand, J.)] when the court speaks of two men, each a perfectionist, independently making maps of the same territory. Though the maps are identical, each may obtain the exclusive right to make copies of his own particular map, and yet neither will infringe the other's copyright.").

"software is governed by some sort of objective methodology according to which programmers simply carry out calculations and writing the correct commands."[52] "A common lay belief is that programming takes place in highly structured environments, relying solely on formal languages and standard techniques, with little or no room for creativity."[53] From this perspective, ""[p]rogramming does not allow personal expression, it is simply a matter of solving mechanic puzzles."[54] "[P]rogramming is regarded as an exclusively instrumental activity, something done just in order to achieve a result: the actual doing is meaningless."[55] It is "something better optimized, minimized, made efficient…. If machines could do it, so much the better: programming in itself is not interesting, only the results are."[56] This "utilitarian perspective reduces the meeting between programmers and software to a mere functional step in the machinery of the software industry."[57] In this context, when the terms "beauty" and "elegance" are applied to source code they are merely synonyms for "speed" and "efficiency."[58] This perception of the programming process colors many court opinions and legal academic writing in this area.[59]

---

52. PIÑEIRO, *supra* note 39, at 31.

53. Aaron Kozbelt et al., *The Aesthetics of Software Code: A Quantitative Exploration,* 6 PSYCHOL. OF AESTHETICS, CREATIVITY, AND THE ARTS 57, 58 (2012).

54. PIÑEIRO, *supra* note 39, at 31.

55. *Id.*

56. *Id.*

57. *Id.*

58. Samuelson, *CONTU Revisited*, *supra* note 8, at 687 ("The creative part of writing a good program may come in finding a faster, more efficient alternative to an obvious but slower way of performing the function."); Burk, *Patenting Speech*, *supra* note 41, at 120 "The goal of a programmer designing software is to achieve functional results in an efficient way. While there may be elements of individual style present in program design, even those style elements concern issues of industrial design, e.g., the choice of one or another programming technique or the clarity (or obscurity) of the functional purpose of a portion of the program.").

59. Comput. Associates Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 704 (2d Cir. 1992) ("The essentially utilitarian nature of a computer program further complicates the task of distilling its idea from its expression."); Ambrose, *supra* note 47, at 17 ("Computer programs…have negligible value outside of their functionality."). Samuelson, *supra* note 8, at 728 ("When Congress decided in 1980 to extend copyright protection to computer programs, it neglected to consider the problems raised by the utilitarian nature of computer programs."); Burk, *Method and Madness in Copyright*, *supra* note 41, at 614 ("For our purposes here this is significant because construction of a device by writing text is equivalent to constructing the device by soldering metal or molding polymers: both will execute processes to produce a determined, monovalent outcome."). Pamela Samuelson*, The Uneasy Case for Software Copyrights Revisited*, 79 GEO. WASH. L. REV., 1746, 1772 (2011) [hereinafter Samuelson, *Uneasy Case*] ("In *Sega Enterprises Ltd. v. Accolade, Inc*. [977 F.2d 1510 (9th Cir. 1992)] the Ninth Circuit embraced Altai's conceptualization of computer programs as utilitarian works that were eligible for only thin copyright protection."); Samuelson et al., *Manifesto Concerning Legal Protection*, *supra* note 40, at 2327 ("Once one understands that programs are machines that happen to have

If this perception were correct, one would expect that now decades into a freestanding software industry, and the innumerable text written attempting to objectify the coding process, the process of programming should be better optimized with programs released on schedule and without functional bugs or vulnerabilities. If true, then a computer should be able to program itself after receiving only the vaguest description of the function it should perform. "Legend has it[, however,] that no programming project has ever been finished on time or within budget, but [there is no need to] go to such an extreme to admit that many applications cost more than planned and arrive late."[60]

If we look at programming from the opposite perspective, however, from the beginning of its conception—the desire for a program to perform a task—to the finished product, we see something very different, *i.e.* a software program from the perspective of a programmer is something very different from the perspective of the end user. It is true that some software engineers view their trade as that of a craft no different from a bricklaying. This view, however, is in the minority.[61]

2. How Programmers Interact with Computer Programs

It is true that "[p]rogrammers write code in order to cause the computer to function in desired ways. But modern computer programs are written in a form, usually textual, that is also meant to be manipulated and understood by human beings. For a programmer to understand what she herself is writing, and to incorporate code that others have written, and to simply learn how to program with greater facility and on a larger more complex scale code has been made legible to people. While a computer system may compile or interpret code, it is important to the nature of code that it is interpreted by

---

been constructed in the medium of text, it becomes easier to understand that writing programs is an industrial design process akin to the design of physical machines."); Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L. J. 1, 7 (1995) ("[C]omputer programs are written for a utilitarian purpose. Expression in the code or structure and organization of a program is normally only incidental to that purpose."); Clark D. Asay, SOFTWARE'S COPYRIGHT ANTICOMMONS 15 (2016) ("Those critiques generally argue that the utilitarian nature of software makes applying traditional copyright law principles to software difficult, and they suggest a variety of legal reforms aimed at better tailoring the law so that it takes into account software's unique functional characteristics.").

60. Piñeiro, *supra* note 39, at 33.

61. Piñeiro *supra* note 39, at 30. "[P]ractitioners have a rich history of discussion and debate of the extent to which computer science is best understood as a science, as engineering, as a craft, or as an art." Aaron Kozbelt, *supra* note 54, at 58.

people as well."[62] "Programmers are extremely conscious of language style, of coding idioms that not only 'get the job done', but do it in a way that is particularly appropriate for that language."[63]

In fact, Donald Knuth, widely considered the founder of computer science as an independent discipline, suggested:

> Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do. There is one very good reason to why code should be readable, namely that it is often the only document that a programmer can resort to when faced with the maintenance of old programs.[64]

In the introduction to his series of books entitled *The Art of Programming,* Dr. Knuth writes "[t]he process of preparing programs for a digital computer is especially attractive because it not only can be economically and scientifically awarding, it can also be an aesthetic experience much like composing poetry or music."[65] Dr. Knuth is known for "teaching [programming] as an art rather than a science."[66] Responding to a question in a lecture he recalled "reading the program SOAP from Stan Poley: 'absolutely beautiful. Reading it was just like hearing a symphony, because every instruction was sort of doing two things and everything came together gracefully.' He also remembers reading the code to a compiler written by Alan Perlis and others: 'plodding and excruciating to read, because it just didn't possess any wit whatsoever. It got the job done, but its use of the computer was very disappointing.'"[67]

"Programmers have a creator-creation relationship to their code, and whether that code fulfils someone else's expectations (users, managers, for instance) is only an aspect of that relationship. Their code reveals a number of things about them, such as their skills, their technical preferences, their beliefs about what can be done with software, in a word, about the kind of programmers they are."[68] "[P]ersonal styles appear at all levels of the programming effort: from those held in less esteem (coding) to those considered truly creative

---

62. Mateas & Montfort, *supra* note 3.

63. *Id*.

64. Donald Knuth, *Literate Programming* (1984), in LITERATE PROGRAMMING 99 (CSLI, 1992).

65. Donald Knuth, *Preface* in THE ART OF COMPUTER PROGRAMMING (1968).

66. FLORIAN CRAMER, WORDS MADE FLESH: CODE, CULTURE, IMAGINATION 27 (Florian Cramer, Piet Zwart Institute 2005).

67. Mateas & Montfort, *supra* note 3.

68. Piñeiro, *supra* note 39, at 24-25.

(software design)."[69] "Programmers respond to software, or relate to it, in aesthetic ways, that is, they do things with it that one only does with aesthetically pleasing objects."[70] While it is true that "beauty" and "elegance" in code can relate to the speed and efficiency of the compiled program, "[s]implicity in programming can be achieved in . . . many different ways."[71] "The fact that beauty appears more quickly discernible [among programmers] than correctness suggests [not only] that aesthetic-laden evaluative processes may drive judgment and decision making about software code" but that there is not a direct correlation between "beauty" and "elegance" of source code and its objective efficiency.[72] "In other words, it may simply be more efficient for programmers to rely on aesthetic intuitions about code than to laboriously understand its details, when writing or revising code."[73]

In sum, while as users of software we view it as utilitarian and function, its creation is not a scientifically objective process, like solving a math problem.[74] "[T]here is no programming methodology today, nor any software development management methodology, that is the undisputed best one, let alone one that could be properly called scientifically correct. Neither the software industry nor the academics know how to make sure programming projects produce the results expected, in the expected time."[75] "[P]rogrammers face different alternatives when they are programming and that these alternatives cannot be, so to speak, calculated away. They require a personal choice from the programmer."[76] "[T]here is no way to know which alternative is the right one, that, in fact, there is no right alternative to choose. All options are equally valid, and the programmer must simply choose one. . . . [T]hese alternatives, even if they are equally

69. *Id.* at 42.

70. *Id*. at 169.

71. *Id*. at 176.

72. Kozbelt et al., *supra* note 53, at 6.

73. *Id*.

74. Since the 1970's work has been ongoing to develop a programming methodology that is as trustworthy as a mathematical theorem. Writing programs that are objectively correct and cannot be hacked. This programming method that incorporates within it proof of correctness is referred to the formal method of programming. Significantly, programs coded in this method tend to be five times larger than programs coded without such a proof of correctness. Researchers in this area conclude that the difficulty in creating error free program and the lack of any objectively correct way to program a task from its technical specification stems from the underlying ambiguity in language itself, which is a necessary burden that any programming language must bear. *See* Kevin Hartnett, *Hacker-Proof Code Confirmed*, QUANTA MAGAZINE (Sept. 20, 2016), http://bit.do/HartnettHackerProof.

75. Piñeiro, *supra* note 39, at 242.

76. *Id.* at 261.

valid from the user's perspective, are not equal to the programmer. Hence, using long variable names, for instance, is not the same as using short variable names, choosing one alternative or the other will not change the functionality of the application but it says something about the programmer."[77] The writerly aspects of programming should not be dismissed. Just as "[n]o literary writer can use language merely as a stopgap device with which to compose an artwork that is not in itself language" writing code is not just the evil necessary to achieve the function, but it is itself the function and something fundamentally different than circuits, gears and levels, which have no ability to do anything other than what they long existed to do.[78]

D. What Becomes a Computer Program

It is important to draw the distinction between computer code—something that simultaneously says something and does it—and programming. The word program comes from "the Greek programma, a written notice to the public. It indicates a procedural way of doing things, which is important for understanding the computational process more broadly: the logic of "if something then something else," for instance.[79] Coding is but a phase of programming process.[80] "[T]he code is the last part of the programmer's action and the first part of the computer's, which makes it the interface between the worlds of human thinking and computer logic."[81]

Programming is an iterative process. "Since the inception of computers (or more broadly, calculating machines), their creators aimed to either solve or accelerate the processes whose 'traditional' implementation would take a lot longer and require considerable resources. However, to make this possible, the thought that it could be done had to come first. This thought had to be turned into an idea—how to achieve this (an algorithm). The idea had to be presented in a form that could be converted into commands for a computer (the source code in a programming language) and then translated into a set of machine statements that a computer could execute (executable program)."[82] Thus, programming has been described as "a human activity that is a great challenge, involving the design of machine

77. *Id.*

78. Florian Cramer, *Exe.cut[up]able statements: The insistence of Code* in CODE: THE LANGUAGE OF OUR TIME 102 (Gerfried Stocker and Christine Schopf, eds., 2003).

79. COX & MCLEAN, *supra* note 4, at 41.

80. Piñeiro, *supra* note 39, at 115.

81. *Id.* at 124.

82. Lula, ET AL., *supra* note 20, at 11.

behavior that can assist, and at times replace, humans in tasks of intellectual nature."[83]

Broadly speaking, the programming process can be broken down into a number of components. The first is the development of an algorithm—creating a method for the task to be performed.[84] This can take the form of a guiding document called a "technical specification," which describe the technical characteristics the program will have.[85] Second is describing this algorithm within a programming paradigm that identifies its architecture from which a programming language can be chosen as well as other decisions can be made such as data structures, identifying the correct libraries of subroutes or classes.[86] Software architecture refers "the organization of a software system as a collection of components, connections between the components, and constraints on how the components interact."[87] Once the design documents mirrors the technical specifications coding can begin.[88] If the design is sufficiently rigorous, then the coding should be nothing more "typing the necessary commands so as to construct the design in a form that a computer (or microprocessor) understanding."[89]

Based on this structure it is easy to see why "[p]rogrammers sometimes give designing a higher status than coding, reasoning that it requires more knowledge to be able to solve the problem (design) than to translate the solution into commands (coding). In fact, occasionally, project organizations differentiate between designers and coders, the former ones generally being better paid. Also, the word 'coders' can be used pejoratively, meaning poor programmers."[90] For our purposes, it becomes important not to elevate the source code itself as the pinnacle of the protected text. We should not discount the architecture and design documents from which route code is written as simply unprotected diagrams of function or structure.

---

83. J-M Hoc ET AL., *Psychology of Programming* 3 (1990).

84. LULA ET AL., *supra* note 20, at 17.

85. PIÑEIRO, *supra* note 39, at 115.

86. LULA ET AL., *supra* note 20, at 17.

87. Jonathan Aldrich, Craig Chambers & David Notkin, *ArchJava: Connecting Software Architecture to Implementation in* PROCEEDINGS OF THE 24TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING 187, 197 (Association for Computing Machinery 2002).

88. PIÑEIRO, *supra* note 39, at 115.

89. *Id*.

90. *Id.*

1. Technical Specifications, Design, and Degrees of Abstraction

Stepping back, it is also helpful to understand what is meant by an algorithm. An algorithm can be defined as "a set of rules, statements, descriptions of successive operations or constraints that determine the sequence of execution of individual operations in order to obtain a certain result."[91] The first requirement of an algorithm is that "the actions to be carried out during the execution of an algorithm cannot contain any ambiguities or default element."[92] The second requirement is that it consist of discrete steps. "A rule can aspire to be an algorithm, if it describes the required actions as a sequence of consecutive basic steps. A 'basic step' means a simple task whose execution does not require any additional explanation."[93] "The third requirement which a recipe has to meet in order to be regarded as an algorithm concerns its finite character. Finite means guaranteeing the completion of the execution of a recipe after a finite number of steps."[94] "The last requirement we set when we want to check that a given recipe should be regarded as an algorithm is a requirement of generality. An algorithm should allow a certain class of problems to be solved or a certain group of goals to be achieved—and not one problem or one goal."[95]

Once the technical specifications are defined for the task, the next step is to translate the algorithm into a task that can be performed by a computer. In this context, the algorithm provides specificity for the task to be performed, but remains several layers of abstraction from any activity that could be performed by a computer. Before discussing the translation of the algorithm into specific software design documents, it might be helpful to outline these levels of abstraction. At its most basic level, "a computer stores instructions and data in memory as a collection of electrical charges."[96] We commonly represent these charges in a binary manner—on and off, or 1 and 0. Each charge is referred to as a bit and a string of four charges is a digit and eight is a byte.[97] Since strings of binary data can quickly

91. LULA ET AL., *supra* note 20, at 18.

92. *Id*. at 19.

93. *Id*. at 20.

94. *Id*. at 24.

95. *Id*. at 26.

96. KIP R. IRVINE, ASSEMBLY LANGUAGE FOR X86 PROCESSORS 9 (Michael Hirsch et al. eds., 6th ed. 2011).

97. *Id*. at 12.

become unwieldly, binary data is more commonly presented as hexadecimal integers that compacts large binary strings.[98]

Now a computer, or more particularly a microprocessor, operates by receiving an input in binary form—for example a key stroke, a mouse click, a sensor trigger, or simply the passing of time, performs a process on that input and provide an output. It is this basic functionality that we can refer to as machine code—or code in its machine language—consisting of the most rudimentary sequence of binary data processed. Now each processor will have its own set of instructions for its basic operation, which is referred to as Instruction Set Architecture ("ISA").[99] Because families of microprocessors will have their own unique ISA, when source code is compiled many times it must be compiled to be compatible with a particular microprocessor's ISA.[100] In this way, to some extent, the same source code can be compiled differently so that it can work with different hardware architectures.[101]

Above the ISA level is a specific program in its assembly language.[102] This is the program in its post-complied state as installed on a system to perform the function of the computer. A program in its assembly language, however, is not strictly binary, but instead "consists of statements written with short mnemonics such as ADD, MOV, SUB, and CALL."[103] Assembly language is the oldest programming language, and of all languages, bears the closest resemblance to native machine language.[104] Specifically, "assembly language has a one-to-one relationship with machine language: Each assembly language instruction corresponds to a single machine-language instruction."[105] An assembler converts the code from assembly language into machine language. While programs can be written in assembly, it is not easy.[106]

---

98. *Id*. at 13.

99. *Id*. at 8.

100. *Id*. at 4 ("Machine language is a numeric language specifically understood by a computer's processor (the PU). All x86 processors understand a common machine language.").

101. *Id.* ("[L]anguage whose source programs can be compiled and run on a wide variety of computer systems is said to be portable. A C++ program, for example, should compile and run on just about any computer, unless it makes specific references to library functions that exist under a single operating system. A major feature of the Java language is that compiled programs run on nearly any computer system.").

102. IRVINE, *supra* note 96, at 8.

103. *Id*. at 4.

104. *Id*. at 2.

105. *Id*. at 4.

106. *Id*. at 525 ("Most programmers do not write large-scale applications in assembly language, doing so would require too much time. Instead, high-level languages hide details that would otherwise slow down a project's development. Assembly language is still used widely,

Above assembly language are higher level languages like C++ that we commonly understand as the language used for writing source code. These higher level languages have "a one-to-many relationship with assembly language and machine language," meaning that "single statement in C++ expands into multiple assembly language or machine instructions."[107] For our purposes, this is relevant because it highlights the difficulty in decompiling a program in its assembly language back to its written source code. Moreover, it underscores the uniqueness of the grammar of a programming language in that it must have a semantical structure that allows it to be understood by programmers and a syntactical structure so that it can be efficiently converted to assembly language and perform desired functionality. Because of the one-to-many relationship between a programming language and its ultimate machine code, all programming languages will necessarily have inefficiencies in their translations. So if performance—the absence of latency between an input and it output—is at a premium, coding as close to the machine level becomes more important.

This hierarchy also helps to emphasize the absence of any one to one relationship between the expression of a function in source code and the ultimate function performed by a computer. Reviewing the assembled code may not necessarily tell you the higher level language in which it was coded.[108] By way of example, an English translation of the Old Testament looks very similar to most people even if the source text for translation was Greek or Hebrew.

Just as source code has a one-to-many relationship with its assembled code, the function reflected by the assembled program can be obtained using a variety of programming languages. This is immediately evident by visiting Rosetta Code website, which provides code to perform the same functionality in as many coding languages as possible.[109] Most programmers will have personal preferences for using particular code for a particular task, but most similarly would agree that the same task can be coded in anyone of the languages that they know. So not only are there many ways to code the same functionality in a given language, there are

---

however, to configure hardware devices and optimize both the speed and code size of programs.").

107. *Id*. at 4.

108. Obviously, this is not the case for languages that do not need to be separately complied in order to perform function on a computer. Such languages are referred to as interpreted programming languages and include JAVA, JAVA Script, Python, Ruby html, and MAX MPS.

109. ROSETTA CODE, http://rosettacode.org/wiki/Rosetta_Code.

innumerable languages that can be used to code that functionality. Even when coding choices become more circumscribed because you are coding for compatibility with a particular system—like the Windows, Apple, android or a flavor of the Linux operating system—it seems somewhat naïve to say a programmer so lacks meaningful choices in how to express the requisite functionality that functionality has merged with the ways of its expression.

Now that we have an understanding of the degrees of abstraction from the creation of the algorithm to the computer actual implementation of the instruction, we can return to the design of a program. Once a programmer has defined the algorithm, the programmer must next define a strategy for implementing the algorithm on a computer—*i.e.*, choose a programing paradigm. A programming paradigm is a way of programming.[110] "We can think of a paradigm as a modeling technique particularly adjusted to problem solving in a computing environment."[111] While a programming paradigm is different than a programming language, different languages lend themselves better to different paradigms. An easy way to understand how programmers write in different paradigms is simply to look at how different paradigms are described:

- **Imperative programming**: defines computation as statements that change a program state
- **Procedural or structured programming**: specifies the steps the program must take to reach the desired state
- **Declarative programming**: defines computation logic without defining its control flow
- **Functional programming**: treats computation as the evaluation of mathematical functions and avoids state and mutable data
- **Object-oriented programming (OOP)**: organizes programs as objects: data structures consisting of datafields and methods together with their interactions
- **Event-driven programming**: the flow of the program is determined by events, such as sensor outputs or user actions (mouse clicks, key presses) or messages from other programs or threads

110. LULA ET AL., *supra* note 20, at 51 ("In case of programming, this term refers to the style of programming used in the given period of development of a computer science or in certain situations. Each paradigm is characterized by a set of mechanisms used by the programmer in the creation of programs and by the method based on which these programs will be executed by the computer.").

111. Georgatos, *supra* note 24, at 10.

- **Automata-based programming**: a program, or part, is treated as a model of a finite state machine or any other formal automaton[112]

While some programming languages lend themselves to multiple programming paradigms, most are optimized to program within a particular paradigm.[113] To understand this let's look at what a program does. Programs compute, they do not deduce. "To compute we start from a given expression and, according to a fixed set of rules (the program) generates a result. . . . To deduce we start from a conjecture and, according to a fixed set of rules (the axioms and inference rules), try to construct a proof of the conjecture. So computation [what the finish program does] is mechanical and requires no ingenuity, while deduction is a creative process."[114] As Dr. Seale thought experiment shows, computers act based on the syntax presented, but they do not understand what they are doing.

Consistent with the way computers operate, programs divide the world into two parts—data and operations on data. "Data is static and immutable, except as the operations may change it. The procedures and functions that operate on data have no lasting state of their own; they're useful only in their ability to affect data."[115] Simply, in response to data received a function is performed causing data to be output.[116]

The programming language C for example is a procedural programming language.[117] "The language may offer various kinds of support for organizing data and functions, but it won't divide the world any differently. Functions and data structures are the basic elements of design."[118] C++, by contrast, is an object oriented programming language, which is a prevalent paradigm for the reasons discussed below. "Object-oriented programming doesn't so much

---

112. *Comparison of programming paradigms,* WIKIPEDIA, http://bit.do/WikiProgrammingParadigms; *Programming Paradigms,* LOY. MARYMOUNT U., http://bit.do/ProgrammingParadigms.

113. Paakki, *supra* note 25, at 197.

114. Frank Pfenning, Lecture for a Carnegie Mellon course in Logic Programming (Fall 2006), http://bit.do/LogicProgramming.

115. Elmar Ludwig, Lecture at University of Osnabrück: Object-Oriented Programming versus Functional Programming: A Comparison of Concepts (2001), http://bit.do/LudwigObjectOriented.

116. LULA ET AL., *supra* note 20, at 126. "A computer program consists of a sequence of statements executed by the computer. The basis for the execution of the program is the input data, which is a source of information. The result of the program operation is the output data, usually stored on a data media. Input/output (I/O) operations are the basis for communication of a computer program with the environment").

117. Ludwig, *supra* note 115, at 3.

118. *Id.*

dispute this view of the world as restructure it at a higher level. It groups operations and data into modular units called objects and lets you combine objects into structured networks to form a complete program. In an object-oriented programming language, objects and object interactions are the basic elements of design."[119]

2. Object-Oriented Programming and Writing Code

While this discussion of programming paradigms may seem like the weeds, object oriented programming paradigms that organize data and function into modules with those modules then able to relate to one another within a structure is important to our discussion. Object oriented programming is not linear, but three dimensional. Rather than compared to building a single structure, object oriented programming is more akin to urban planning where different structures are needed with different tasks expected to occur in each with means for data to move between structures, but not always in the same order or by the same route. This paradigm not only allow for the creation of extremely complex programs and allows numerous programmers to simultaneously work on a program, but it also allows modules build for one program to be reused in different programs assuming the data and function of the module can communicate with the modules of the new city (continuing our urban planning analogy).

For a given module to be added to a structure it must be able to communicate with the data and modules in the new architecture. This is not an issue of programming language—simply because a module is written in C++ does not mean it can meaningfully communicate with any other module written in that language. While not completely analogous, this is somewhat akin to idioms of speech. People in the United States and the UK both speak English, but many times meaning is lost where a word or phrase to a Londoner invokes a different meaning of function to an American, like when asked in a British sandwich shop whether you want salad on your sandwich.

So let's step back again. Source code written in a language like C++ or JAVA can be written on a piece of paper. It can also be written on Notepad which is found on any Widows based computer and compiled from there to become a functioning program. Typically, however, programs are written using Software Developer Kits (SDK), which provides a number of building block that help expedite the software development process, including frameworks, and libraries of common tasks along with a debugger (finding errors in your code), a

119. *Id.*

compiler and other tools. An Integrated Development Environment (IDE) collects the tools of SDK within a single Graphic User Interface (GUI). To utilize the shared functionality requires that the programmer conform to naming convention the SDK's Application Program Interface (API), *i.e.* if a programmer want to utilize routine functionality the code must call the functionality by its right name. Using our analogy, if I ask for salad on my sandwich in London it will come with lettuce and tomato, but the likely response at a Subway in Los Angeles would be a confused stare. This is precisely what was at issue in *Oracle America, Inc. v. Google, Inc.*[120] By copying how routine tasks are called when using JAVA's SDK, programmers already familiar with programming in JAVA could easily create applications for Google's android operating system in the android SDK. by changing how the routine task is performed, however, android avoids the "write once run anywhere" requirement for a royalty free license to Oracle's copyright on the JAVA programming language.[121]

Utilizing an object-oriented programming paradigm and utilizing accepted naming conventions allow functionality written for one application to be used in another. One of the most ubiquitous examples of such a module is MySQL, an open source database utilized in innumerable free and commercial programs.[122] Oracle, which owns MySQL, distributes the program under GNU General Public License (GPL).[123] This licenses entities and individuals to royalty-free use of MySQL. If, however, MySQL is made part of a publicly distributed program, the license requires that the source code for the module, including any modification be made publicly available.[124] This requirement, however, does not preclude an owner from selling a program that utilizes MySQL.[125] If a user does not want to conform to the GNU GPL, Oracle offer other licensing options, but they are not royalty-free.[126] The GNU GPL is one of several software licensing options available that utilize intellectual property law to accomplish the goals of Free and Open Source Software movement,

120. *Oracle*, 750 F.3d at 1347.

121. The ability of Copyright to protect a language is a topic beyond the scope of this article.

122. *MySQL*, ORACLE, http://bit.do/MySQLdatabase.

123. *Commercial License for OEMs, ISVs and VARs*, ORACLE, http://bit.do/MySQL-License; *Frequently Asked Questions about the GNU Licenses*, FREE SOFTWARE FOUNDATION, http://bit.do/WhatDoesGPLStandFor.

124. *MySQL*, *supra* note 123.

125. *Frequently Asked Questions about the GNU Licenses*, *supra* note 123.

126. *MySQL*, *supra* note 123.

and are commonly referred to as copyleft licenses.[127] The alternative, less "political" licensing scheme are "attribution only" licenses like the one governing the Apache, the most widely used webserver program.[128] Google distributes android under the Apache 2.0 license that allows anyone to "take android, significantly modify it, and not release the source code to others."[129]

What is relevant here to our discussion is the norm within programming of using existing tools to speed up routine processes and the willingness of programmers to make available the source code of tools they have developed that might ease the burden on their toiling brethren. The intent and expectation of making source code freely available is not only that the tool will be helpful to the community, but also that the community will review comment and improve the tool for the greater good. The website GitHub is a good example of such a repository of code.[130]

E. The Community of Programmers

The community underpinning associated with coding is one reason for long confrontational relationship between programmers and intellectual property law. A bit of history may be helpful here. Historically software and hardware were tied together, as they remain with Apple.[131] Many tie the development of an independent software industry to the time when Bill Gates and Paul Allen broke with IBM to make Microsoft an independent company writing software for a generic computer and not simply programs for personal computers

---

127. Johan Soderberg, HACKING CAPITALISM THE FREE AND OPEN SOURCE SOFTWARE MOVEMENT 19 (2008) ("The politics of the hacker movement gravitate around the issue of public access to source code…. Software released under Free and Open Source Software (FOSS) licenses are required to be published together with the source code. In proprietary software the source code is hidden away as unintelligible binary code."); Asay, SOFTWARE'S COPYRIGHT ANTICOMMONS, *supra* note 59, at 18 ("These licensing mechanics thus helped spread the FOSS movement's norms of collaboration and software freedom, which norms were in some ways a natural fit within a software industry increasingly devoted to object-oriented programming and its building-block approach to software development.").

128. Soderberg, *supra* note 127, at 24 ("Apache is a software program for running web servers. As of January 2006 it held 70% of the market. The largest commercial competitor, Microsoft, had only 20% of the market.").

129. Clark D. Assay, *A Case for the Public Domain*, 73 OHIO ST. L. J. 1, 28 (2013).

130. GITHUB, http://bit.do/GitHub ("How people build software. Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.").

131. Samuelson, *Uneasy Case*, *supra* note 59, at 1750-51 (discussing Stephen Breyer's then 40 year old article questioning the prudence of copyright protection for software at a time when software was typically tied to a unique customer's singular application).

manufactured by IBM.[132] For our purposes, however, the history of the Unix operating system is more relevant.[133]

UNIX was developed in the late 1960s early 1970s by AT&T engineers during their free time.[134] These engineers consulted extensively with professors at University of California at Berkeley. AT&T obtained a copyright in UNIX and then licensed its use for a nominal fee to Berkeley as well as a number of other public and private entities.[135] Thereafter, as a result of an antitrust consent decree with the Department of Justice, AT&T agreed refrain from pursuing any computer-related businesses. During this period, the professors at Berkeley, their students, and other enthusiasts continued to develop and refine UNIX cooperatively.[136] In 1982, when the restrictions imposed by the consent decree were lifted, AT&T attempted to enforce its rights in UNIX.

> The attempt by AT&T to privatize UNIX qualifies as one of the most notorious enclosures saluting the dawn of the 'information age'. It had a resolute impact on the collective mindset of the programmers' community and fueled their skepticism towards big corporations and the intellectual property regime. AT&T's bid to own UNIX demonstrated that copyright can be used to rob authors of their work, the very opposite of the ideological justification for the law. Hackers thus realized that the collective authorship of software developers has to be shielded from the legal powers instituted in a single party by copyright law.[137]

In response to AT&T efforts, programmers worked to strip all remnants of AT&T's UNIX operating system from their current UNIX flavor, and this spawned the formation of the Free Software Foundation.[138] The Foundation's version of UNIX called GNU, (recursively) standing for "GNU's Not UNIX."[139] The need for legal

---

132. Soderberg, *supra* note 127, at 18.

133. CRAMER, *supra* note 37, at 121 ("The Unix operating system which runs on almost any kind of hardware is a prime example of software as an abstraction from hardware."); Cox, *supra* note 4, at 21 ("Behind this crucial issue of access to information is the history of the sharing of source code, itself rooted in the history of the UNIX operating system and its precarious position between promises of the public domain and commercial enterprise, corresponding to the differences between free software and open source development.").

134. Unix System Laboratories v. Berkeley Software, 832 F. Supp. 790, 794 (D. NJ 1993).

135. *Id.* at 793.

136. Soderberg, *supra* note 127, at 19 ("[I]n 1982, AT&T was relieved from the anti-trust ruling, which had prevented the company from entering the computer business. The company soon began to enforce ownership rights over UNIX. By then the operating system had been extended and rewritten many times over by students, scientists and enthusiasts collaborating across institutions and corporations").

137. *Id*.

138. *Id.* at 23.

139. *Id*. at 19.

protection for their shared work to ensure that their work remained open and free was quickly realized when a programmer who had contributed code to this version sold his code to a third party who decided to assert ownership over the code.[140] This "experience contributed to the creation of the General Public License. The nickname for GPL is telling; 'Copyleft-All Rights Reversed.'"[141] In words of its founder, Richard Stallman: "Copyleft uses copyright law, but flips it over to serve the opposite purpose: instead of a means of privatizing software, it becomes a means of keeping software free."[142]

Over the years, many versions of the UNIX operating system have been released, the most famous being LINUX, which Linus Torvald single handedly started in his "garage."[143] Shortly thereafter, programmers began to flock to LINUX.[144] This spawned the continuing battle between computer programs as property and software as a service. The debate about whether the public good is best served by proprietary programs built with capital as opposed to open source programs written with the passion of their programmers continues today.[145] What is important for our purpose, is that both proprietary and freely distributed programs need rules covering their use in order to further their political end, and today those rules mostly come from the Copyright Act. The distinction between the two regimes is the ability of the programming community to look under the hood—view the source code that achieves the functionality. Errors in programming are endemic and can lead to untold problems. Here, the open source movement has the advantage as evident from their refrain "given enough eyeballs, all [software] bugs are shallow."[146]

Finally, the application of intellectual property law, and in particular copyright law, alone does not tip the balance in favor of some over others. A program or even a programming language, no matter its merit, has no value until it is adopted and used. The decisions by Oracle to continue to distribute JAVA under a royalty-

---

140. *Id*.

141. *Id*.

142. Soderberg, *supra* note 127, at 19.

143. *Id*. at 23.

144. *Id*.

145. *Id.* at 26 ("Linus Torvald's have [sic] offered his own explanation to the GNU/Linux phenomenon. The competitive edge of free software over proprietary software owes to the higher motivation of its authors. Speaking at a Linux User Group meeting in San Francisco, he stated: 'Those other operating systems aren't bad because of [technical detail] A or technical detail B. Those systems are bad because the people don't care[.]'").

146. Clark D. Asay, *A Case for the Public Domain*, 73 OHIO ST. L.J., no. 10 (2013), at 1, 14.

free license, and similarly for Microsoft to make its Visual Basic IDE, available royalty free are driven in large part to promote their use and maintain the viability of the programming language they each support. Someone could conceive and create the most efficient programming language of all time, but if its use is conditioned on a restrictive and expensive license, its adoption would be unlikely. This balance is obviously grasped by the owners of some of the most successful game apps, which make them free to play until you are hooked and they then start charging.

The enforcement of intellectual property laws, or the lack thereof, cannot be seen as the reason for the success of any language, operating system, or program over another. The best example of this fact is Microsoft, which, with the adoption of its operating system, began giving away application that could run on that system to capture the word processing, spreadsheet, graphics and browser application market. The risk to competition greater than the inefficient application of intellectual property law to computer programs is the network itself. Like in computer hardware, software today is built upon standards. While those who control the foundational elements of these standards, like a system's operating system, are highly motivated for third parties to write applications to make they systems more desirable to consumers—think of the IOS or android—the complexity of these systems themselves give their owners enormous power over economy. Think what would happen if Microsoft decided to patch all versions of its operating system with code that turned the system off permanently. Today's intellectual property laws cannot require Apple, Microsoft or Google to continue to make available their operating systems upon which so many other systems depend, but existing law intellectual property laws effectively prevent the creation of perfect substitutes to protect the public good if they decided to turn them off.[147]

II. COPYRIGHT LAW

A. Congress May Enact Laws To Promote Science and the Useful Arts

147. Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1253 (3d Cir. 1983) ("Franklin claims that whether or not the programs can be rewritten, there are a limited 'number of ways to arrange operating systems to enable a computer to run the vast body of Apple-compatible software', Brief of Appellee at 20. . . . Franklin may wish to achieve total compatibility with independently developed application programs written for the Apple II, but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged.").

Copyright protection can be simply summarized as granting rights for how something is expressed in exchange for what is expressed. Stated another way, copyright does not protect what you teach, but how you teach it.[148]

Copyright protection as we know it is not guaranteed by the United States Constitution.[149] Instead the founding fathers saw the utility of affording Congress the power "[t]o promote the Progress of Science and useful Arts[] by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries,"[150] believing that "[t]he States cannot separately make effectual provisions for [the protection of authors or inventors]."[151] Congress' exercised its right "[t]o Promote the Progress of Science[[152]] by securing for . . . Authors the exclusive Right to their Writings" by enacting our current copyright scheme, and Congress acted "[t]o Promote the Progress of useful Arts by securing for . . .

148. Brief English Sys. v. Owen, 48 F.2d 555, 556 (2d Cir. 1931) ("Once concede that the defendant Owen could lawfully write, or write about, any system of shorthand his ability permitted and his book has nothing of consequence in common with what is covered by the plaintiff's copyrights. The manner of treatment is substantially dissimilar and original. Without proof of the kind of appropriation mentioned above, the plaintiff has no cause of action.").

149. Goldstein v. California, 412 U.S. 546, 555 (1972) ("The clause thus describes both the objective which Congress may seek and the means to achieve it.").

150. U.S. CONST. art. I, § 8, cl. 8. While early on some questions existed about whether purely aesthetic works fell within the meaning of this constitutional provision, the Supreme Court resolved that issue in Mazer v. Stein, 347 U.S. 201, 211 (1953). Similarly, thereafter, courts have repeatedly judged themselves incompetent to judge the quality or creativity of an aesthetic work. *See* Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 582-83 (1994).

> As Justice Holmes explained, '[i]t would be a dangerous undertaking for persons trained only to the law to constitute themselves final judges of the worth of [a work], outside of the narrowest and most obvious limits. At the one extreme some works of genius would be sure to miss appreciation. Their very novelty would make them repulsive until the public had learned the new language in which their author spoke.

Id. (quoting Bleistein v. Donaldson Lithographing Co., 188 U.S. 239, 251 (1903) (circus posters have copyright protection)); *cf.* Yankee Publ'g Inc. v. News Am. Publ'g, Inc., 809 F. Supp. 267, 280 (S.D.N.Y. 1992) (Leval, J.) ("First Amendment protections do not apply only to those who speak clearly, whose jokes are funny, and whose parodies succeed.") (trademark case).

151. THE FEDERALIST NO. 43 (James Madison). Four years before the ratification of the Constitution, the Continental Congress passed a resolution recommending that the several states enact laws securing to authors and publishers of new books the "exclusive right of printing, publishing and vending" such new book for a period of up to 28 years. 8 JOURNALS OF CONGRESS: CONTAINING THEIR PROCEEDINGS FROM NOV. 2, 1782 TO NOV. 1, 1783 (Folwell's Press 1800).

152. "The 'Progress of Science' . . . refers broadly to 'the creation and spread of knowledge and learning.'" Golan v. Holder, 565 U.S. 302, 324 (2012); *see* Lawrence B. Solum, *Congress' Power to Promote the Progress of Science: Eldred v. Ashcroft*, 36 LOY. L.A. L. REV. 1, 47-53 (2002).

Inventors the exclusive Right to their Discoveries" by adopting our patent laws.[153]

Consistent with the constitutional intent, the period of exclusivity granted authors and inventors is subsidiary to the public good the laws are designed to foster, *i.e.*, a means to a public good end.[154] It has long been recognized that "[t]he sole interest of the United States and the primary object in conferring the monopoly, lie in the general benefit derived by the public from the labors of authors. Copyright, like a patent is 'at once the equivalent given by the public, benefits bestowed by the genius and mediation and skills of individuals and the incentive to further efforts for the same import objects.'"[155]

Over time, Congress has amended both of these schemes as it sees fit to promote dissemination of learning, knowledge and discoveries for the public good. If Congress ever deemed either or both of these schemes inadequate, it has the constitutional authority to scrap the scheme and start anew so long as such scheme includes provisions "securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[156]

B. Copyright's Bargained-For Exchange: Protecting "How You Teach" in Exchange for "What You Teach"

Currently, the two schemes differ somewhat in the manner in which each promotes the public good. Recognizing the value and importance of prompt public dissemination of discoveries, the patent law includes a bargained-for exchange whereby inventors agree to

---

153. 2 WILLIAM F. PATRY, PATRY ON COPYRIGHT, § 3:4 at 3-12.1 (West 2015) (Although "[t]he clause uses neither 'copyright' nor 'patent' . . . science is joined with authors and writings, while useful arts is joined with inventors and discoveries."); *Golan*, 565 U.S. at 324 ("Perhaps counterintuitively for the contemporary reader, Congress' copyright authority is tied to the progress of science; its patent authority to the progress of useful arts.").

154. Dowling v. United States, 473 U.S. 207, 216 (1985) ("In contrast, the Government's theory here would make theft, conversion, or fraud equivalent to wrongful appropriation of statutorily protected rights in copyright. The copyright owner, however, holds no ordinary chattel."); Twentieth Century Music Corp. v. Aiken, 422 U.S. 151, 156 (1975) ("The immediate effect of our copyright law is to secure a fair return for an author's creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good."); *Darden v. Peters,* 488 F.3d 277, 284 (4th Cir. 2007) ("Copyright is solely a creature of statute; whatever rights and remedies exist do so only because Congress provided them. *See* Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417, 431 (1984). Thus, as there is no constitutional right to copyright registration, the Register's refusal to register Darden's claim cannot be "contrary to constitutional right" as it must be for section 706(2)(B) to apply.").

155. Fox Film Corp. v. Doyal, 286 U.S. 123, 128 (1932) (quoting Kendall v. Winsor, 62 U.S. 322, 327-28 (1856)).

156. U.S. CONST. art. I, § 8, cl. 8.

publicly disclose their inventions shortly after first exploitation in exchange for the exclusive right to make, sell or use their invention for a specified period of time. Significantly, federal law does not foreclose common law protections for inventors unwilling to participate in this exchange, allowing them to maintain the secrecy of their inventions and prosecuting and recovering damages against those who misappropriate the secrets of their invention. [157]

As is the case in patent law, Congress' original formulations of federal copyright law limited protection to works presented to the public, leaving common law protections available for unpublished works.[158] But unlike patent law, Congress did not circumscribe the time in which the creator needed to publish after creation. Following endless contortions to define what constituted a publication in differing media, Congress, in its 1976 Copyright Act, scrapped the publication requirement, but not out of any reconsideration of the importance of creators making their work available to the public.[159] Even without the express publication requirement, the Supreme Court continues to recognize that copyright's overriding purpose is to induce the dissemination of ideas and expression.[160]

---

157. Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470 (1974) (state trade secret law can coexist with federal patent law); Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 146 (1989) ("As we have noted in the past, the Clause contains both a grant of power and certain limitations upon the exercise of that power. Congress may not create patent monopolies of unlimited duration, nor may it 'authorize the issuance of patents whose effects are to remove existent knowledge from the public domain, or to restrict free access to materials already available.'" (quoting Graham v. John Deere Co. of Kansas City, 383 U.S. 1, 6 (1966)).

158. *See* L. Ray Patterson, *Free Speech, Copyright, and Fair Use*, 40 VAND. L. REV. 1, 2 (1987). Arguably, the difficulty in recognizing what constituted a publication with newer technologies was an impetus for dispensing with the publication requirement. *Id.*

159. Curiously, the Supreme Court, construing the 1976 Copyright Act in *Eldred v. Ashcroft*, 537 U.S. 186 (2002), cited that lack of a public disclosure requirement as a meaningful reason to distinguish its patent jurisprudence from copyright law. *Id.* at 217. In rejecting appellant's effort to analogize the two schemes, the Court stated "patents and copyrights do not entail the same exchange," indicating that its "references to a *quid pro quo* [that] typically appear in the patent context" are not applicable to copyright. *Id.* at 216. The *Eldred* Court states "[f]or the author seeking copyright protection, in contrast, disclosure is the desired objective, not something exacted from the author in exchange for the copyright." *Id.* The *Eldred* Court continues by drawing the distinction with patent law that "copyright gives the holder no monopoly over knowledge," while "[t]he grant of a patent does prevent full use of being made of knowledge." *Id.* at 217. While the *Eldred* Court acknowledges that the copyright holder obtains something less than a monopoly over his creation, the Court does not view this as ceding the idea underlying their expression to the public as *quid pro quo* for the protection copyright affords expression. The Supreme Court's notion that somehow there is no public dedication of the work as a *quid pro quo* seems inconsistent with its jurisprudence that First Amendment protection extends not only to the ability to speak but afford a citizen a constitutional right of access to information and the dissemination of ideas. *See* Patterson, *supra* note 158, at 2.

160. *Cf.* Golan v. Holder, 565 U.S. at 326 (2012) ("Evidence from the founding, moreover, suggests that inducing *dissemination*—as opposed to creation—was viewed as an appropriate

This distinction between patent and copyright protection, which is a holdover from the publication requirement for federal protection, creates one of the greatest inefficiencies with applying copyright protection to computer software. As currently constructed, someone can disseminate their code and have it widely adopted into larger systems only later to assert rights and force users into a hostage situation whereby the users must either rewrite their code to perform the functionality[161] without its expression or pay a royalty. The Supreme Court's recent decision holding that latches cannot be used to shorten the Congressionally mandated limitations period for a copyright claim makes this risk more significant.[162]

By requiring creators of expressive works to register their works within a particular time of public dissemination and identify the limitations on use (*e.g.*, free to use with attribution, private, or paid royalty requirement), the public knows the restrictions on their use of the expression, which furthers the public good. Like all forms of expression, price determines use. Many of us watch movies on Netflix, paid for by our monthly subscription that we would never separately pay for to watch in a theater. In the same way, when alternative source code is available to perform a particular function, differences in costs and restrictions will play a role in which code is adopted.

In addition to express time limitations, Courts have long recognized limitations to authors' granted monopoly rights under copyright law in order to ensure that the law conformed to its overarching purpose of furthering public good.[163] As Justice Brandeis observed nearly 100 years ago in *International News Service v.*

---

means to promote science.").

161. The ability to meticulously change expression to adopt an expression's idea without its function is well recognized in copyright outside the protection for computer programs. Paycom Payroll, LLC v. Richison, 758 F.3d 1198, 1205 (10th Cir. 2014) ("A corollary of the 'sameness' requirement is that '[c]opying deleted or so disguised as to be unrecognizable is not copying.'" (quoting See v. Durang, 711 F.2d 141, 142 (9th Cir. 1983)).

162. Petrella v. Metro-Goldwyn-Mayer, Inc., 134 S. Ct. 1962, 1977 (2014) ("Congress' time provisions secured to authors a copyright term of long duration, and a right to sue for infringement occurring no more than three years back from the time of suit. That regime leaves 'little place' for a doctrine that would further limit the timeliness of a copyright owner's suit.").

163. United States v. Paramount Pictures, Inc., 334 U.S. 131, 158 (1948) ("The copyright law, like the patent statutes, makes reward to the owner a secondary consideration."); Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417, 477 (1984) ("There are situations, nevertheless, in which strict enforcement of this monopoly would inhibit the very 'Progress of Science and useful Arts' that copyright is intended to promote.";) Mathews Conveyer Co. v. Palmer-Bee Co., 135 F.2d 73, 85 (6th Cir. 1943) ("fair use . . . may be made of copyrighted material based on the principle that subsequent workers in the same field are not to be deprived of all use thereof . . . otherwise, the progress of science and the useful arts would be unduly obstructed.").

Associated Press: "The general rule of law is, that the noblest of human production knowledge, truths ascertained, conceptions, and ideas become, after voluntary communication to others, free as the air in common use."[164] Early court decisions formulated a distinction between what is expressed and how it is expressed, which many times is given the short hand idea/expression dichotomy.[165] Recognizing the paramount purpose of the public good, copyright, in a manner similar to the patent scheme, encourages and rewards authors for their creations by allowing them to profit from their original expression of their creations in exchange for leaving to the public good the use what has been expressed. Judge Learned Hand, discussing the idea/expression dichotomy, stated that a plaintiff's "copyright did not cover everything that might be drawn from her play; its contents went to some extent into the public domain. We have to decide how much, and we are aware as anyone that the line, wherever it is drawn, will seem arbitrary, that it no excuse for not drawing it."[166]

This distinction long recognized by the court between what is expressed and how it is expressed is codified as part of the 1976 amendment in Section 102(b) of the Copyright Act. Importantly, and consistent with existing jurisprudence, Congress codified that "what is expressed" is much broader than simply an "idea," but also includes the underlying "procedure, process, system, method of operation, concept principle, or discovery, regardless of the form in which it is described, explained, illustrated or embodied in such work."[167] Contrary to the conclusion of some court and commentators, Section 102(b) is not intended to demarcate copyright from patent protection[168] or to enact an exclusion from what

164. Int'l News Servs. v. Associated Press, 248 U.S. 215, 250 (1918) (Brandeis, J., dissenting).

165. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Shakespeare's 'ideas' [are no more] capable of monopoly [than] Einstein's Doctrine of Relativity, or Darwin's theory of the Origin of Species.").

166. *Id*. at 122.

167. 17 U.S.C. § 102(b) (2012); *Golan*, 565 U.S. at 328 ("The idea/expression dichotomy is codified at 17 U.S.C. § 102(b)[.]"); *see* M. Kramer Mfg. Co., Inc. v. Andrews, 783 F.2d 421, 434 (4th Cir. 1986) ("In effect, what Congress intended in this subsection was simply a codification of the statement made by the Supreme Court in *Mazer v. Stein*, 347 U.S. at 207 that copyright protection extended 'only to the expression of the idea—not the idea,' thus distinguishing between 'idea' and 'expression,' between 'methodology or processes' and 'writings,' and, in effect, between patentability and copyrightability.).

168. Lexmark Intern. v. Static Control Components, 387 F.3d 522, 534 (6th Cir. 2004) ("This provision embodies the common-law idea-expression dichotomy that distinguishes the spheres of copyright and patent law."); Samuelson, *Functionality and Expression*, *supra* note 50, at 18 ("A third proposition about § 102(b) that should be uncontroversial is that procedures, processes, systems, and methods of operation are excluded from the scope of copyright protection in part to maintain a balance between the role of copyright in protecting authorial

otherwise would be protected by copyright law.[169] Simply, at the time Section 102(b) was enacted it was well understood that like copyright, "one may not patent an idea."[170] As stated in the Committee Notes accompanying this subsection "'Section 102(b) . . . in no way enlarges or contracts the scope of copyright protection under the present law. Its purpose is to restate, in the context of the new single federal system of copyright, that the basic dichotomy between expression and idea remains unchanged.'"[171] As applied to software protection, Congress continued "'Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.'"[172]

Importantly, this distinction has never been limited to aesthetic ideas and their expression.[173] In *Mazer v. Stein,* the Supreme Court stated that it found "nothing in the copyright statute to support the argument that the intended use or use in industry of an article eligible for copyright bars or invalidates its registration. We do not read such a limitation into the copyright law."[174] In fact, two-thirds of the original scope of copyright protection as enacted in 1790 consisted of maps and charts, which "are valued to the extent they offer useful organizations of facts."[175]

---

expression and the role of patent law in providing protection to inventions in the useful arts.").

169. *Oracle*, 750 F.3d at 1339 (rejecting Google's argument "there is a two-step copyrightability analysis, wherein Section 102(a) grants copyright protection to original works, while Section 102(b) takes it away if the work has a functional component.").

170. Gottschalk v. Benson, 409 U.S. 63, 71 (1972).

171. M. Kramer Mfg. Co., Inc. v. Andrews, 783 F.2d 421, 434 (4th Cir. 1986) (quoting H.R. REP. NO. 94-1476, at 5670 (1976)).

172. *Id.* at 434-35 (quoting H.R. REP. NO. 94-1476, at 5670 (1976)).

173. "Blueprints for large buildings (more committee work), instruction manuals for repairing automobiles, used car value guides, dictionaries, encyclopedias, maps . . . are routinely copyrighted, and challenges to the validity of these copyrights are routinely rejected." American Dental Ass'n v. Delta Dental Plans Ass'n, 126 F.3d 977, 978 (7th Cir. 1997) (citing Educational Testing Services v. Katzman, 793 F.2d 533 (3d Cir. 1986) (Scholastic Aptitude Test); CCC Information Services, Inc. v. Maclean Hunter Market Reports, Inc., 44 F.3d 61 (2d Cir. 1994) (list of used-car prices); Lipton v. Nature Co., 71 F.3d 464 (2d Cir. 1995) (terms describing groups of animals). See also 17 U.S.C. § 101 (including "architectural plans" within the definition of "pictoral, graphic and sculptural works" that are copyrightable); PAUL GOLDSTEIN, COPYRIGHT: PRINCIPLES, LAW AND PRACTICE § 2.15.2 (1989) (discussing copyright protection for computer programs)); *see also* General Drafting Co. v. Andrews**,** 37 F.2d 54, 55 (2d Cir. 1930) ("Automobile maps similar to those in suit are clearly the subject of copyright..."); Edwin K. Williams & Co. v. Edwin K. Williams & Co, 542 F.2d 1053, 1060 (9th Cir. 1976) (affirming protection of instruction in accounting book).

174. Mazer v. Stein, 347 U.S. 201, 218 (1954).

175. *American Dental Ass'n*, 126 F.3d at 978-79 (recognizing that a taxonomy of dental procedures and nomenclature is entitled to copyright protection). The original codification of the

The fact that the use of a machine is necessary for expression has long been held to be irrelevant.[176] Copyright jurisprudence has rarely hesitated to recognize that Congress intended its statute to reach any form of expression, whether fanciful, factual, practical, or scientific. By not reaching the idea itself or the expressed "procedure, process, system, method of operation, concept, principle, or discovery," a copyright does not protect an article's usefulness as a "procedure, process, system, method of operation, concept, principle, or discovery."[177] Moreover, with respect to physical articles, if the article's usefulness cannot be separated from creative expression, then no copyright protection would attach—think of a design for a chair or an article of clothing.[178] An apt example is *Brandir Intern., Inc. v. Cascade Pacific Lumber Co*., where an artist modified an existing sculpture to make it more compatible as a bike rack.[179] While the original sculpture was copyrightable, the modified version was not because it had become simply a product of industrial design where utility could no longer be separated from its creative expression.[180] By contrast, where a decorative sculpture could be separated from its function as a lamp, the sculpture itself could still be protected by copyright.[181]

Significantly, this required demarcation between expression and usefulness is limited in the Copyright Act to articles that are, or incorporate, "[p]ictorial, graphic, and sculptural works."[182] "Such works" are protected by copyright "only if, and only to the extent

---

Copyright Act is at 1 Stat 124.

176. Burrow-Giles Lithographic Co. v. Sarony, 111 U.S. 53, 59 (1884) (extending copyright protection to photography).

177. 17 U.S.C. § 102(b).

178. Carol Barnhart Inc. v. Economy Cover Corp., 773 F.2d 411, 414 (2d Cir. 1985) ("Since the four Barnhart forms [mannequins] are concededly useful articles, the crucial issue in determining their copyrightability is whether they possess artistic or aesthetic features that are physically or conceptually separable from their utilitarian dimension."); Galiano v. Harrah's Operating Co., Inc., 416 F.3d 411, 422 (5th Cir. 2005) ("Gianna makes no showing that its designs are marketable independently of their utilitarian function as casino uniforms."); Samuelson, *Functionality and Expression*, *supra* note 50 ("The overall design of a chair or automobile, by contrast, may well have an aesthetic character, but the expression in these creations cannot be separated (that is, they are merged) from their utilitarian functions.").

179. Brandir Int'l, Inc. v. Cascade Pacific Lumber Co., 834 F.2d 1142 (2d Cir. 1987).

180. *Id.* at 1147 ("Had Brandir merely adopted one of the existing sculptures as a bicycle rack, neither the application to a utilitarian end nor commercialization of that use would have caused the object to forfeit its copyrighted status. Comparison of the RIBBON Rack with the earlier sculptures, however, reveals that while the rack may have been derived in part from one of more 'works of art,' it is in its final form essentially a product of industrial design.").

181. *Mazer*, 347 U.S. at 217 (holding that statute that was part of a lamp was copyrightable).

182. 17 U.S.C. § 101.

that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of exiting independently of, the utilitarian aspects of the article."[183] Some argue that an ability to separate expression from utility should extend to computer programs as well, which would effectively render programs unprotectable because they both do (utility) and say (express).[184] Neither the Copyright Act nor case law, however, require such an explicit demarcation between utility and expression for non-objects like literary works or motion pictures. A simple example proves this point. School districts chose algebra textbooks because of their perceived utility—the ability to teach students algebra. While such a textbook is incapable "of existing independently of" the algebra it teaches, no one disputes that the algebra book is protected by copyright. Instead, copyright protection would extend to how it teaches algebra even though how it teaches is completely interwoven with what it teaches. This same reasoning must apply to computer programs. While copyright does not protect the functions and processes performed by a program, the fact remains that the manner in which a program instructs the computer to perform such a function or process cannot exist independently from the function or process performed. A program simultaneously says what it does and does it, but this does not mean that the creative expression of a program is not protected.

The seminal case in this area is *Baker v. Selden*,[185] over which much ink has already been spilt. Simply, the plaintiff in *Baker* held a copyright in a book explaining a bookkeeping method and claimed blank forms that allowed persons to practice the method described in the book—although not similar to those shown in the book—infringed the book's copyright. The Supreme Court summarized the case as follows:

> The defendant uses a similar plan so far as results are concerned; but makes a different arrangement of the columns, and uses different headings. If the complainant's testator had the exclusive right to the use of the system explained in his book, it would be difficult to contend that the defendant does not infringe it, notwithstanding the difference in his form of arrangement; but if it be assumed that the system is open to public use, it seems to be equally difficult to contend that the books made and sold by the defendant are a violation of the copyright of the complainant's book

183. *Id.*

184. Gordon, *supra* note 34.

185. Baker v. Selden, 101 U.S. 99 (1880).

> considered merely as a book explanatory of the system. [186]

Stated simply, the issue confronted by *Baker* is whether a copyright protects not only how something is taught but also grants rights over the subject of the teaching.[187] The *Baker* Court held it did not. "The description of the art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself. The object of the one is explanation; the object of the other is use. The former may be secured by copyright. The latter can only be secured, if it can be secured at all, by letters-patent."[188] By way of example,

> [t]ake the case of medicines. Certain mixtures are found to be of great value in the healing art. If the discoverer writes and publishes a book on the subject (as regular physicians generally do), he gains no exclusive right to the manufacture and sale of the medicine; he gives that to the public. If he desires to acquire such exclusive right, he must obtain a patent for the mixture as a new art, manufacture, or composition of matter.[189]

To state another way, "[f]ew 'how-to' works are 'systems' in Baker's sense. If they were, architectural blueprints could be freely copied, although the Berne Convention Implementation Act of 1988, Pub. L. 100-567, 102 Stat. 2854, adds protection for "architectural plans" to the statute. Descriptions of how to build or do something do not facilitate monopoly of the subject-matter being described, so the concern of *Baker* is not activated."[190]

Commentators debate whether *Baker* is the foundation of the idea/expression dichotomy, even though the *Baker* Court does not use that phraseology.[191] Similarly, other have argued that the copyright limitation described in *Baker* reflects patent law exclusive authority over process, methods, and compounds. A simpler explanation is possible. Copyright protects creative expression and is unconcerned

---

186. *Id.* at 100.

187. *Id.* at 101.

188. *Id.* at 105.

189. *Id.* at 102-03. Similarly, the *Baker* Court continues, "The copyright of a work on mathematical science cannot give to the author an exclusive right to the methods of operation which he propounds, or to the diagrams which he employs to explain them, so as to prevent an engineer from using them whenever occasion requires." *Id.* at 103.

190. *American Dental Ass'n,* 126 F.3d at 981.

191. William Patry, *Copyright and Computer Programs: A Failed Experiment and a Solution to a Dilemma*, 46 N.Y. L. SCH. L. REV. 201, 216 (2002) ("The origin of the idea-expression dichotomy is frequently traced to the Supreme Court's 1880 decision in Baker v. Selden. It is questionable whether the origin is correctly ascribed: the opinion never refers to ideas and was decided on the ground of lack of originality.").

with the uniqueness of what is expressed. This is why books about the battle at Gettysburg, Einstein's theory of relativity, how to use Windows 10, as well as translation of an ancient text into modern English receive copyright protection.[192] In each the foundational facts, methods, discoveries, ideas, and processes were already known. Copyright law reflects the policy decision that to reward authors for their original expression of these known ideas serves the public good. [193] Stated another way, "Einstein's articles laying out the special and general theories of relativity were original works even though many of the core equations, such as the famous $E=mc^2$, express 'facts' and therefore are not copyrightable. Einstein could have explained relativity in any of a hundred different ways; another physicist c[an] expound the same principles differently."[194]

C. Copyright and Code

Returning to code, any number of expressions of source code can result in the same functionality—maybe even result in identical object code to achieve that function. The fact that source code is directed to functionality—the same way a work of history is directed to facts or an algebra textbook is directed to algebra—does not deny its author a copyright to how she expresses that source code to achieve that functionality.[195] Or stated another way, "'[t]hat the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability.'"[196] Specifically, as will be developed below, the ability of a programmer to look at functionality, observe its structure as it functions, and write her own source code based on the observed structured functionality without infringing the copyright in the underlying computer program does not

---

192. *See* Feist Publications, Inc. v. Rural Telephone Service Co., 499 U.S. 340, 345-46 (1991) ("Originality does not signify novelty; a work may be original even though it closely resembles other works so long as the similarity is fortuitous, not the result of copying. To illustrate, assume that two poets, each ignorant of the other, compose identical poems. Neither work is novel, yet both are original and, hence, copyrightable.").

193. The Supreme Court has referred to copyright protections as "the engine of free expression." Harper Row Publishers, Inc. v. Nation Enterprises, 471 U.S. 539, 558 (1985).

194. *American Dental Ass'n*, 126 F.3d at 979.

195. *Apple Computer, Inc. v. Formula Intern., Inc.,* 562 F. Supp. 775, 782 (C.D. Cal. 1983) ("As the CONTU Report aptly put it, 'One is always free to make the machine do the same thing as it would if it had the copyrighted work placed in it, but only by one's own creative effort rather than by piracy.'") (quoting *Final Report of the National Commission on New Technological Uses of Copyrighted Works*, p. 21 (1978) republished in 3 COMPUTER L.J. 51 (1981)).

196. *Apple Comput.*, 714 F.2d at 1252 (quoting *Final Report of the National Commission on New Technological Uses of Copyrighted Works*, p. 21 (1978) republished in 3 COMPUTER L.J. 51 (1981)).

mean that the source code itself and its structure are not protected by copyright law.[197] Here, we must be precise. Like a novel or a film, copyright protection of a computer program is not limited to simply syntactical copying, but instead should found in the identity in structure rather than simply identity of word choices.[198]

Copyright law grants authors certain exclusive rights to original expression[199] of a creative idea. Broken down, copyright law protects expression that is original, regardless of whether the facts, ideas, processes, or methods described are unique. [200] The time and expense

197. Hutchins v. Zoll Medical Corp., 492 F.3d 1377, 1383 (Fed. Cir. 2007) ("[C]opyright does not protect the technologic process independent of the program that carries it out; that is, the copyright covers the way the process is described in the written or electronic form of the computer program, but does not cover the process independent of the copyrighted program.").

198. Many courts and commentators argue that the copyright protection for computer programs should be limited to syntactical identity—what they refer to as "literal" infringement—copying the ordered words themselves. *Comput. Associates*, 982 F.2d at 712 ("[It] may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes."). This conclusion is simply contrary to how copyright law is applied in other contexts. For example, other literary works in the Ninth Circuit are evaluated based on an "extrinsic" and "intrinsic" test. "The 'extrinsic test' is an objective comparison of specific expressive elements. '[T]he test focuses on articulable similarities between [between structural elements like] the plot, themes, dialogue, mood, setting, pace, characters, and sequence of events in two works.'" Cavalier v. Random House, Inc., 297 F.3d 815, 822 (9th Cir. 2002) ((quoting Kouf v. Walt Disney Pictures & Television, 16 F.3d 1042, 1045 (9th Cir. 1994) (quotation marks and citation omitted). "The 'intrinsic test' is a subjective comparison that focuses on "whether the ordinary, reasonable audience" would find the works substantially similar in the 'total concept and feel of the works.'" *Id.* (quoting *Kouf,* 16 F.3d at 1045 (quotation marks and citation omitted)). By comparison lists of literal similarities "between the two works are "inherently subjective and unreliable," particularly where the list contains random similarities, and many such similarities could be found in very dissimilar works." Herzog v. Castle Rock Entertainment, 193 F.3d 1241, 1257 (11th Cir. 1999) (quoting Beal v. Paramount Pictures Corp., 20 F.3d 454, 460 (11th Cir. 1994)); Williams v. Crichton, 84 F.3d 581, 590 (2d Cir. 1996) ("[S]uch lists are 'inherently subjective and unreliable," particularly where "the list emphasizes random similarities scattered throughout the works.') (quoting Litchfield v. Spielberg, 736 F.2d 1352, 1356 (9th Cir. 1984).

199. Originality is distinct from the novelty requirement under patent law. In patent law, an invention or method of use is denied protection if it is not truly new, *i.e.* novel. For example, a patent can be invalidated if, through the combination of several pieces of prior art, the invention or method would be obvious to anyone knowledgeable in that field even if the inventor could prove that they were the first to combine these elements. *KSR*. By contrast, the creator of the film *Alien v. Predator* is not denied a copyright even if it could be proved that it would be obvious to any sci-fi film buff to combine these two characters in a movie and express the film as it was ultimately made. While the copyright holder arguably must procure the rights to the characters of Alien and Predator for use in his film, he would have no copyright in the prior works as they contributed to his film, the copyright holder could enforce his copyright as to the original expression that flows from the combination of these two works.

200. *Mazer*, 347 U.S. at 214 ("They must be original, that is, the author's tangible expression of his idea."). Secondly, copyright law does not afford protection for all original expressions of ideas but only those that are creative. *Feist Publications, Inc.*, 499 U.S. at 350-51 ("Facts, whether alone or as part of a compilation, are not original and therefore may not be copyrighted. A factual compilation is eligible for copyright if it features an original section or

of compiling data alone does not warrant protection under copyright law. Similarly, a mannequin that simply mimics a human body as a method for displaying clothing is not protected for its useful purpose, but to the extent that its creator adds unique features the mannequin is protected even though useful.[201] Copyright law reaches only the original expression of any creative effort and not the effort itself.

This protection of original expressions irrespective of the uniqueness of the underlying idea can easily be juxtaposed with the protections afforded under the patent law. Patent law gives inventors rights to tangible use of inventions and discoveries. Like copyright law, patent law does not afford protection over an idea or discovery itself, but their application or use, *i.e.* their function.[202] This explains why patent protection is not available for an abstract idea, a mathematical formula or a discovery in nature untethered to a unique application or use. [203] Unlike copyright law, patent law focuses on the uniqueness of the invention described in an application rather than the originality of how the invention is described in the application.

Thus, rather than affording protections on some sort of continuum, patents and copyrights protect fundamentally different things. This could not be more stark than in the Copyright Acts protection of computer software as the protection of a literary work.

---

arrangement of facts but the copyright is limited to the particular selection or arrangement. In no event may copyright extend to the facts themselves."); Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947) (data recorded by a machine not copyrightable).

201. Compare Pivot Point v. Charlene Products, Inc., 372 F.3d 913, 931 (7th Cir. 2004) ("Applying this test to the Mara mannequin, we must conclude that the Mara face is subject to copyright protection. It certainly is not difficult to conceptualize a human face, independent of all of Mara's specific facial features, i.e., the shape of the eye, the upturned nose, the angular cheek and jaw structure, that would serve the utilitarian functions of a hair stand and, if proven, of a makeup model.") with Carol Barnhart Inc. v. Economy Cover Corp., 773 F.2d 411, 418 (2d Cir. 1985) ("Applying these principles, we are persuaded that since the aesthetic and artistic features of the Barnhart forms are inseparable from the forms' use as utilitarian articles the forms are not copyrightable.").

202. *Apple Comput.*, 714 F.2d at 1253 ("'Just as a patent affords protection only to the means of reducing an inventive idea to practice, so the copyright law protects the means of expressing an idea; and it is as near the whole truth as generalization can usually reach that, if the same idea can be expressed in a plurality of totally different manners, a plurality of copyrights may result, and no infringement will exist.'" (quoting Dymow v. Bolton, 11 F.2d 690, 691 (2d Cir. 1926)) (emphasis excluded)).

203. Alice Corp. Pty. Ltd. v. CLS Bank Intern., 134 U.S. 2347 (2014) ("The 'abstract ideas' category embodies 'the longstanding rule that "[a]n idea of itself is not patentable."'" (quoting Gottschalk v. Benson, 409 U.S. 63, 67 (1972) (quoting Rubber-Tip Pencil Co. v. Howard, 80 U.S. 498, 507 (1874)))); Le Roy v. Tatham, 55 U.S. 156, 175 (1853) ("A principle, in the abstract, is a fundamental truth; an original cause; a motive; these cannot be patented, as no one can claim in either of them an exclusive right."); Dan L. Burk, *Patenting Speech*, 79 TEXAS L. REV. 99, 137 (2000) ("In patent, as in copyright, it has long been blackletter law that ideas are not patentable, but their physical embodiments are.").

Simply, a patent is not available for text.[204] Moreover, there is simply not a place where on original creative expression of a function ends and the function itself begins.[205] Instead, if there is just one or a few practical ways of expressing something, there is little, if any, creativity separating the function from its expression, *i.e.* ways of listing names in a phone book. Some refer to this as the merger doctrine, where the mode of expressing a fact, idea, process or function is one and the same with the fact, idea, process or function itself.

D. Access to the Text of a Computer Program

Copyright law does not protect a work itself, but protects against another copying the expression of a work.[206] Stated simply if someone breaks into an author's home and steals their original manuscript or other work of art, that theft is not copyright infringement.[207] Similarly, and unlike patent law, if someone completely cut off from all of society pens *Harry Potter and the Sorcerer's Stone* without any knowledge or access to the original work, the identity between the two works would not matter, assuming, as strange as it seems, that the

---

204. In re Russell, 48 F.2d 668, 669 (CCPA 1931) ("The mere arrangement of printed matter on a sheet or sheets of paper, in book form or otherwise, does not constitute 'any new and useful art, machine, manufacture, or composition of matter,' or 'any new and useful improvements thereof,' as provided in section 4886 of the Revised Statutes, 35 USCA § 31."); Pamela Samuelson, The Strange Odyssey of Software Interfaces and Intellectual Property Law, UC Berkeley Recent Work, 5 (Dec. 12, 2008), http://bit.do/StrangeOdyssey ("Odyssey Patent law had long excluded 'printed matter,' such as texts, from the scope of its protection, even though printed matter is a manufactured artifact, and hence literally within the meaning of the term 'manufacture,' one of the four categories of inventions for which patents may issue."); Burk, *Patenting Speech*, *supra* note 41, at 141. ("U.S. courts have long held that mere arrangement of lines or symbols does not constitute patentable subject matter as such markings would not be within patentable subject matter as a new and useful machine, manufacture, or composition of matter.").

205. The fact that within an article function and the expression of function never meet is different than the fact two works can have similarity of function and expression. When addressing similarity of function and express the two exist on a continuum. Specifically, two works can share an identity of function but express that function differently. They can also share similarity of function and have various degrees of similarity of how they each express the function.

206. Perris v. Hexamer, 99 U.S. 674, 675-76 (1879) ("A copyright gives the author or the publisher the exclusive right of multiplying copies of what he has written or printed. It follows that to infringe this right a substantial copy of the whole or of a material part must be produced.").

207. *Cf.* Stephens v. Cady, 55 U.S. 528, 531 (1853) ("The copperplate engraving [of a map], like any other tangible personal property, is the subject of seizure and sale, on execution, and the title passes to the purchaser, the same as if made at a private sale. But the incorporeal right, secured by the statute to the author, to multiply copies of the map, by the use of the plate, being intangible, and resting altogether in grant, is not the subject of seizure or sale by means of this process — certainly not at common law.").

author of the second work had no ability to copy the original work. Both would be entitled to a copyright because both would be original within the meaning of copyright—they were both independently created.[208]

Thus to plead and prove copyright infringement the copyright holder must prove first that the alleged infringer had access to the protected elements of the work and that protected elements were copied. Obviously, one cannot copy what one cannot see. Unlike almost any other literary work protected by copyright, proof that a defendant possessed a computer program is not proof that such person has access to the "set of statements or instruction to be used directly or indirectly in a computer in order to bring about a certain result" as the protection of a computer program as a literary work is defined.[209] Without question the complete copying of an executable program that will allow the protected program to run on another computer necessary copies not only the functions of the program but its expression—its original statement or instructions—that cause the function. [210] The bundle of rights afforded to an author's original expression of a creative idea by copyright, however, are not limited to verbatim copying of an entire work.[211]

As we discussed above, programs coded in a language that requires compiling to function on a particular computer—such as a program coded in C++ as opposed to a program coded in an interpreted language like JAVA—translates these instructions into object and assembly code. As discussed above, there is no one-to-one

---

208. Mazer v. Stein, 347 U.S. 201, 217-18 (1954) ("[T]wo men, each a perfectionist, independently making maps of the same territory. Though the maps are identical, each may obtain the exclusive right to make copies of his own particular map, and yet neither will infringe the other's copyright."); Feist Publications, Inc. v. Rural Telephone Service Co., 499 U.S. 340, 345 (1991) ("Original, as the term is used in copyright, means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity."); L. Batlin & Son, Inc. v. Snyder, 536 F.2d 486, 490 (2d Cir. 1976) ("Originality is, however, distinguished from novelty; there must be independent creation, but it need not be invention in the sense of striking uniqueness, ingeniousness, or novelty, since the Constitution differentiates 'authors' and their 'writings' from 'inventors' and their 'discoveries'.").

209. 17 U.S.C. § 101 (1980).

210. Belford v. Scribner, 144 U.S. 488, 508 (1892) ("The rule is well settled, that, although the entire copyrighted work be not copied in an infringement, but only portions thereof, if such portions are so intermingled with the rest of the piratical work that they cannot well be distinguished from it, the entire profits realized by the defendants will be given to the plaintiff.").

211. Ty, Inc. v. Publications Int'l, Ltd., 292 F.3d 512, 518-519 (2002) (recognizing that pictures of Beanie Babies were a derivative work and that a catalogue that included such pictures needs to be evaluated based on the infringement of the derivative work rather than against the dolls themselves).

correlation between a line of assembly or object code with the generating source code and the same object or assemble code can be generated from source code instruction coded in the same or different programming languages. While the assembly code and object code of the compiled program are accessible and can be copied, if one is copying something less than the entire program it is extremely difficult to know precisely what one is copying. This is Dr. Seale's thought experiment discussed above, the assembly and object code have "meaning" to a computer solely due to their syntactical structure—*i.e.* the computer acts based on the structure of the commands received not based on any actual understanding of what the structure means. In most instances, however, the grammar of assembly and object code do not impart any meaning to us. It is not that assembly and object code are devoid of semantics, but that the semantics are hidden by symbols that we do not readily understand. No different than if someone handed to us a book in Chinese when we do not speak the language or a musical score when we do not read music. No one would contend that the book or the score are devoid of expression. Instead, we simply do not understand the expression. We do not have access to it.

An apt example *is Atari Games Corp. v. Nintendo of America Inc*.[212] In *Nintendo,* Atari wanted to sell games that worked with Nintendo game system, but did not want to pay Nintendo a license fee required to make its games compatible. Atari was able to isolate the object code within the game system that controlled the locking system, but was unable to design a key simply from the object code. It was only when Atari wrongfully obtained the source code for the Nintendo system from the Copyright Office that Atari was able to correct the errors in its code and gain compatibility of its games to the system.[213] The Federal Circuit ultimately affirmed the district court grant of a preliminary injunction in Nintendo's favor, affirming that Nintendo's compatibility code was protectable as expression, and that Atari did not have a fair use defense because it did not lawfully possesses a copy of Nintendo's accessible expression—the source code from which it created compatibility.[214]

---

212. *Atari*, 975 F.2d at 835.

213. *Id.* at 836 ("After obtaining the 10NES source code from the Copyright Office, Atari again tried to read the object code from peeled chips. Through microscopic examination, Atari's analysts transcribed the 10NES object code into a handwritten representation of zeros and ones. Atari used the information from the Copyright Office to correct errors in this transcription.").

214. *Id.* at 844 ("To invoke the fair use exception, an individual must possess an authorized copy of a literary work . . . Because Atari was not in authorized possession of the Copyright Office copy of 10NES, any copying or derivative copying of 10NES source code

Curiously, despite the fact that *Nintendo* is a fairly early case in the jurisprudence of copyright protection of software, neither courts nor commentators have picked up on this issue of access as it relates to computer programs. Instead, both seem to simply gloss over the issue. The impact of this blind spot is probably best seen in commentators' challenge that the Federal Circuit's recent decision in *Oracle America, Inc. v. Google, Inc*.[215] In particular, this impact relates to the argument that the Federal Circuit's decision in *Oracle* cannot be reconciled with the First Circuit's earlier decision in *Lotus Development Corp. v. Borland Intern. Inc.*

In *Oracle,* Google admitted to copying portions of the application programming interface, or API, of the software development kit, or SDK, Oracle distributed to assist in writing code in the JAVA programming language.[216] Google had access to and copied source code. APIs allow programmers to use short phrases—declaring code—to call routine programming commands to help expedite the programming process. Google admittedly copied the command language known to anyone who programs in JAVA, but changed the underlying instruction called so that what was programmed would be uniquely functional in its android platform. This allowed programmers experienced with JAVA to easily code in android, but would result in a program compatible only with android. The District Court concluded that what Google copied was simply functional structure of the JAVA SDK and that its system and method of operation was not protectable under copyright law, citing *Lotus*.[217] The Federal Circuit reversed, noting that "while the defendant in *Lotus* did not copy any of the underlying code, Google concedes that it copied portions of Oracles declaring source code verbatim."[218] While much more can be said about *Oracle*, let's stop and focus on this point of distinction.

Commentators challenge that the Federal Circuit's decision in *Oracle* that copyright protection of a computer program can extend to the structure of its source code—including structures that easy programming in a coding language—runs contrary to precedent, and in particular the First Circuit's well-respected *Lotus* decision.[219] In

---

from the Copyright Office does not qualify as a fair use.").

215. *Oracle*, 750 F.3d at 1370.

216. *Id.* at 1376.

217. *Id.* at 1364.

218. *Id.*

219. Gordon, *supra* note 34, at 2. ("Debates over the conflict between Oracle and Lotus have largely ignored a middle road that supports the Lotus result without the potential for

Lotus, Borland intentionally mimicked the menu command hierarchy displayed to users of Lotus' spreadsheet program so that users could easily transition to Borland's competing Quatro Pro spreadsheet program.[220] Significantly, "[i]n so doing, Borland did not copy any of Lotus's underlying computer code."[221] The First Circuit reversed the district court's grant of a permanent injunction, "hold[ing] that the Lotus menu command hierarchy is an uncopyrightable 'method of operation;" *i.e.* function rather than expression.[222] As an unprotected function, "[t]he fact that Lotus developers could have designed the Lotus menu command hierarchy differently is immaterial . . . ."[223]

Based on our discussion, the distinction between *Oracle* and *Lotus* could not be more apparent. Borland never had access to the Lotus source code that achieved the functions observed in the functioning of the Lotus spreadsheet. What was copied was a function without knowledge of the instruction that achieved that function. By contrast, Google has access to the JAVA SDK source code and copied the source code to achieve the requisite functionality. Specifically, Google was found to have sufficiently copied the instruction to achieve the function. *Lotus* is simply not a case about copyright protection of a computer program within the meaning of the Copyright Act, which focuses on the "set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[224] Instead *Lotus* is better categorized as a case concerned with the protection of "audiovisual works," which are works that consist of a series of related images . . . intended to be shown by the use of machines."[225] There should be no dispute that the copying of observed function with no access to the expression used to instruct the computer to achieve that function is not copying relevant to the Copyright Act. Even if by some chance of fate, the code Borland used to mimic Lotus' functionality turned out to be identical, there still would be no copyright infringement of the computer program because Borland did not have access to the source code.

---

overkill some observers see in Lotus. This middle road is a doctrine known as the 'explanation/use' distinction. Laid out in the classic Supreme Court case of *Baker v. Selden* (1800), and ratified by statutory provisions of the Copyright Act including the much-ignored Section 113(b), the 'explanation/use distinction.'").

220. Lotus, 49 F.3d at 810 ("Borland included in its Quattro and Quattro Pro version 1.0 programs 'a virtually identical copy of the entire 1-2-3 menu tree.'" (quoting Lotus Dev., Inc. v. Borland Int'l Inc., 831 F. Supp. 202, 212 (D.Mass. 1983))).

221. *Id.*

222. *Id.* at 815.

223. *Id.* at 816.

224. 17 U.S.C. § 101.

225. *Id.*

While the manner in which information is presented to a user can be protected by copyright—as is the case with computer games[226]—the First Circuit decision that the manner in which Lotus presented its menus to user for use of its spreadsheet program says nothing about whether copyright protects how an author structures a program to achieve functionality.[227]

The lack of access to a program's source code, however, is not in and of itself determinative on whether an alleged infringer has access to the literary work that is a computer program protectable by the Copyright Act. While meaning as opposed to the syntax of assembly and object code may be hidden, that does not mean assembly and object code themselves are simply functional and devoid of semantics as some commentators have argued.[228] The fact that an infringer many not know the expression they are taking and may care only about the function expressed rather than how it is expressed—as in the case of copying an entire executable file—does not change the fact both the function and how the function has been expressed has been copied. But in deciding whether the protected "instructions to be used . . . in a computer . . . to bring about a certain result" have been infringed, as opposed to simply the visual presentation of the program itself, the courts should consider the extent to which the accused infringer had access to the protected instructions.

---

226. Tetris Holding, LLC v. Xio Interactive, Inc., 863 F. Supp. 2d 394, 410 (D.N.J. 2012) ("While there might not have actually been 'literal copying' inasmuch as Xio did not copy the source code and exact images from Tetris, Xio does not dispute that it copied almost all of visual look of Tetris. This leaves one to wonder what Xio believed was protectable expression. After the purportedly careful analysis it undertook to understand copyright law, apparently Xio believed it could engage in wholesale copying of the Tetris look and relate almost every visual element to a 'rule,' finding none of Tetris's visual expression copyrightable.").

227. Samuelson, *CONTU Revisited*, *supra* note 8, 681 ("What a program displays is not at all the same thing as the instructions which cause the display to occur. That is, it is not the program instructions that are displayed on the screen when a program is being executed. In general, one cannot get the program instructions to be -displayed on the screen even if one wants to. Of course, for the most part, the computer user does not want to know what the program instructions 'say' to the computer. The user does not care how the program does what it does, just that it does what it is supposed to do.").

228. *See id*. ("There are several responses to this contention. First, while source code is clearly a literary work within the meaning of the statute, machine code is not. To say that a machine-readable program is a literary work because it is a 'copy' of a literary work, namely the source code, is like saying that a building is a drawing because the architectural plans which were used to make it are drawings. It is only if we do not understand what programs are that we can consider a machine code a literary work. A second response is that of John Hersey: 'To call a machine-control element a copy of a literary work flies in the face of common sense.'" (quoting Dissent of Commissioner Hersey, Final Report of the National Commission on New Technological Uses of Copyrighted Works 21 (1978) republished in 3 COMPUT. L.J. 51 (1981)).

As discussed above, simply observing the visual functionality of a computer program in operation at best disclose the algorithm that preceded the architectural and functional specifications used to code what would become a computer program. Such algorithms are akin to "what is being taught" and is distinct from the structural and explanatory documents from which source code is created. Thus observing the visual display of a program's function is at least a step removed both the syntax and semantics that state to the programming community the function to be performed (semantics) and instruct the computer the function to be performed (syntax).

The fact that the semantics of object code are hidden and not removed is demonstrated in the Ninth Circuit's decision in *Sega Enterprises Ltd. v. Accolade, Inc*.[229] In *Sega*, like *Nintendo*, a game manufacturer wanted their game to be accessible to a game platform without paying for access to the platform.[230] Rather than surreptitiously obtaining the game systems source code, Accolade disassembled the object code from the system and loaded it onto a computer to identify the sequence of instructions necessary for compatibility.[231] As the name suggest, disassembly refers to a method of attempting to divine source code from object code. When Accolade succeeded and began releasing unlicensed game for Sega system, Sega sued and obtained a preliminary injunction, which the Ninth Circuit reversed.[232] The Ninth Circuit rejected the notion that copying object code for purposes of identifying its functionality was not actionable copying within the meaning of the Copyright Act.[233] The Ninth Circuit reaffirmed that copyright protection reached the specific instructions to a computer in object code. "Although some scholarly authority supports that view, we have previously rejected it based on the language and legislative history."[234] The Ninth Circuit was not persuaded that object code that controlled interoperability was purely functional and beyond the reach of the Copyright Act.[235] The Ninth Circuit, however, agreed with Accolade that Accolade's copying to learn how to achieve interoperability although infringing was a fair use. "We conclude that where the disassembly is the only way to gain access to the ideas and functional elements embodied in a computer program and where there is a legitimate reason for seeking such

229. Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).
230. *Id.* at 1514.
231. *Id.* at 1515.
232. *Id.* at 1515-16.
233. *Id.* at 1518-19.
234. *Id.* at 1519.
235. *Sega Enters*, 977 F.2d at 1519.

access, disassembly is a fair use of a copyrighted work as a matter of law."[236] As will be developed below, however, copying for purposes of understanding function as held in *Sega* is distinct from copying in order to achieve interoperability.[237]

E. While Substantial Similarity Is a Surrogate for Copying, Substantiality Should Not Be a Requirement for Infringement

Once access is possible, copying is possible. For our purposes there are two issues: What was copied, and what is the proof that it was copied? Let's address an issue with respect to the latter question first to resolve some confusion. Rarely, is there ever evidence of direct copying.. For this reason, courts allow juries to presume copying relevant to trigger infringement from evidence that the defendant had access to the work and that the challenged work bears a "substantial similarity" to the protected work in a meaningful manner.[238] Courts typically speak of proof of access and copying as being on a sliding scale meaning the more compelling the evidence of similarity the less proof of access is necessary.[239] This presumption of copying created by proof of access and substantial similarity can be rebutted by the defendant by proof of independent creation. Unlike in patent law, a shared identity between two works alone, even if indistinguishable, does not require a finding of infringement.[240]

---

236. *Id.* at 1526.

237. *Oracle*, 750 F.3d at 1370 ("We disagree with Google's suggestion that *Sony* and *Sega* created an "interoperability exception" to copyrightability."); *Apple Comput*., 714 F.2d at 1353 ("Franklin may wish to achieve total compatibility with independently developed application programs written for the Apple II, but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged.").

238. Atari, Inc. v. North American Philips Consumer Electronics Corp., 672 F.2d 607, 614 (7th Cir. 1982) ("Because direct evidence of copying often is unavailable, copying may be inferred where the defendant had access to the copyrighted work and the accused work is substantially similar to the copyrighted work."); Novelty Textile Mills v. Joan Fabrics Corp., 558 F.2d 1090, 1092 (2d Cir. 1977) ("Since direct evidence of copying is rarely, if ever, available, a plaintiff may prove copying by showing access and "substantial similarity" of the two works.").

239. Three Boys Music Corp. v. Bolton, 212 F.3d 477, 485 (9th Cir. 2000) ("[s]ubstantial similarity is inextricably linked to the issue of access. In what is known as the "inverse ratio rule," we "require a lower standard of proof of substantial similarity when a high degree of access is shown." (quoting Smith v. Jackson, 84 F.3d 1213, 1218 (9th Cir. 1996)).

240. Selle v. Gibb, 741 F.2d 896, 901 (7th Cir. 1984) ("Proof of copying is crucial to any claim of copyright infringement because no matter how similar the two works may be (even to the point of identity), if the defendant did not copy the accused work, there is no infringement.").

While substantial similarity must be proven in the absence of direct evidence of copying, the question remains, "substantial as to what?" For example, if two four-hundred-page books share shocking similarity with respect to a single ten-page chapter, is that sufficient similarity to form the basis of a copyright claim? The confusion seems to stem from two distinct uses of the term "substantial" both of which are relevant to copyright law, but for different purposes. The substantiality part of the surrogate test for copying focuses on whether there is sufficient identity between the two works to allow a jury to presume that the defendant copied expression in the protected work sufficient to switch the burden to the defendant to prove that the work was independently created. The other use of substantial refers to whether the defendant copied a sufficient modicum of the protected expression to warrant action under the copyright law. In the absence of direct evidence of copying, the distinction between these two uses of substantial is blurred because the fewer elements a challenged work shares with the protected work, even if that identity is substantial, tends to tip against a finding that a presumption of copying is warranted.

The importance of copyright law's distinctive use of "substantial" becomes clear when there is direct evidence of copying. For example, what if a teenage character in my fantasy novel travels from New York to London in one chapter to have lunch with the film and literary character Harry Potter? While this encounter may only occupy five pages of my three-hundred-page book, even if it is necessary to my story that the reader understand that my character met with *the* Harry Potter, I would still need to admit to direct copying of the Harry Potter character. But could a court determine that my use of Ms. Rowling's expression too insubstantial to allow her to prove infringement irrespective of the fair use defense?

While the Copyright Act does not articulate any quantitative minimum threshold of actual copying necessary to support an infringement claim, some courts turn to the legal maxim "*'de minimus no curat lex'* (sometimes rendered, 'the law does not concern itself with trifles") [as] insulat[ing] from liability those who cause insignificant violations of the rights of other, so as to avoid the "perplexing task" of applying the fair use factors.[241] In fact, some

241. Ringgold v, Black Entertainment Television, Inc., 126 F.3d 70, 74, 76 (2d Cir. 1997). While *Ringgold* has been cited by several other cases as supporting a quantitative threshold for infringement, the Second Circuit in *Ringgold* noted the absence of case law in this area. *Id.* at 74 (noting that the *de minimus* threshold was crossed and reversing the district court's grant of summary judgment based on fair use); *Gordon v. Nextel Communications*, 345 F.3d 922, 925 (2003) (fleeting out of focus images of illustrations in commercial were *de minimus*). More

argue that the various substantial similarities tests employed by the differing jurisdictions not only considers substantiality for the purposes of determining whether copying of expression can be inferred, but necessarily consider substantiality for the purposes of determining whether a meaningful amount of expression has been copied.[242] The Federal Circuit in *Oracle* avoided the issue of a free standing *de minimis* defense, finding what Google admitted to copying to be substantial.[243]

Burdening the copyright holder with proving the substantiality of what was taken, however, makes little sense and has been roundly rejected by courts, at least in relation to music sampling.[244] If reason does not support requiring a copyright holder to meet the multifaceted substantial similarity test where copying is conceded, then

---

curious is the Ninth Circuit's use of *de minimus* in the *Newton v. Diamond*, 388 F.3d 1189, 1196 (9th Cir. 2004). In *Newton*, the Ninth Circuit affirmed the grant of summary judgment that the copying of the composition copyright in a rap song was *de minimus* because it was nothing more than a "trite, and generic three-note sequence" and the average audience would not discern that Newton's hand as a composer, apart from his talents as a performer, from the Beastie Boy use of the sample." *Id.* (the Beastie Boys had a license to the sound recording from the record label). The Ninth Circuit's qualitative evaluation of Newton's contribution as a composer and its evaluation of whether listeners would discern his compositional contribution is unusual.

242. Mark A. Lemley, *Our Bizarre System for Proving Copyright Infringement* (Aug. 18, 2010). Stan. Public Law Working Paper No. 1661434, http://bit.do/LemleyBizarreSystem ("For copying to rise to the level of infringement, it must include more than *de minimis* amount of copyrighted expression."); Situation Management Systems, Inc. v. Asp. Consulting LLC, 560 F.3d 53, 58 (1st Cir. 2009) ("Our analysis, therefore, focuses entirely on whether the district court properly applied the second element of the *Feist* test. That prong 'itself involves two steps: the plaintiff must show (a) that the defendant actually copied the work as a factual matter, ... and (b) that the defendant's `copying of the copyrighted material was so extensive that it rendered the infringing and copyrighted works substantially similar.'"'' (quoting T-Peg, Inc. v. Vt. Timber Works, Inc., 459 F.3d 97, 108 (1st Cir. 2006)); Society of Holy Transfiguration v. Gregory, 689 F.3d 29 (1st Cir. 2012) ("A party seeking to establish copying of a work's original elements must make a dichotomized showing. The copyright holder first must factually establish, whether via direct or circumstantial evidence, that the alleged infringer copied the protected work. Secondly, the holder must show that the copying was so flagrantly extreme that the allegedly infringing and copyrighted works were, for all intents and purposes, 'substantially similar.'") (citations omitted).

243. *Oracle*, 750 F.3d at 1378 ("Oracle argues that the Ninth Circuit does not recognize a de minimis defense to copyright infringement and that, even if it does, we should affirm the judgments of infringement on grounds that Google's copying was significant. Because we agree with Oracle on its second point, we need not address the first, except to note that there is some conflicting Ninth Circuit precedent on the question of whether there is a free-standing *de minimis* defense to copyright infringement or whether the substantiality of the alleged copying is best addressed as part of a fair use defense.").

244. Bridgeport Music, Inc. v. Dimensions Films, 410 F.3d 792, 802 (2005) ("[E]ven when a small part of a sound recording is sampled, the part taken is something of value. No further proof of that is necessary than the fact that the producer of the record or the artist on the record intentionally sampled because it would (1) save costs, or (2) add something to the new recording, or (3) both."). While some argue that sampled music should be treated differently than other forms of copyright infringement, there is no cogent reason for doing so.

substantiality should mean nothing more in a circumstantial case than sufficient identity in some protectable expression to allow a finding of copying to be inferred irrespective and not to support some quantitative threshold.

Rather than imposing on the copyright holder the burden of showing that what was taken was substantial, better reasoned cases hold insubstantial copying, while still infringing, does not warrant injunctive relief and supports the affirmative defense of a fair use.[245] Even where the defendant is unable to make the case for a fair use of the protected expression, the public good is served by denying injunctive relief but awarding damages where the copying is insubstantial when compared with defendant's original expression a as a whole.[246] As will be discussed below, while the quantitative and qualitative use made of a protected expression is relevant to the affirmative defense of fair use, its application is not limited to quantitatively insubstantial use.

F. A New Work That Is Less Than a Facsimile of the Work As a Whole Is a Derivate Work

It stands to reason that proof of any copying of an original expression of a creative idea, whether by direct evidence or by proof of access and substantial similarity, is infringing. Are all infringing works necessarily derivative works? The most common conceptions of a derivative works are collections of several works—*i.e.* collections of poems or short stories—or transfers of a work to a different medium –*i.e.* books to movies, movies to plays, animation to merchandise. Section 103(a) of the Copyright Act, however, simply defines a derivative work as "a work based upon one or more pre-existing works." That a derivative work includes "any . . . form in which a work may be recast, transformed or adopted," indicates a legislative intent that a derivative work must include some modicum of original expression of a creative idea, distinguishing a derivative work from a mere copy.[247] Key again is a creativity. An artist who

245. *See* Mattel, Inc. v. MGA Entertainment, Inc., 616 F.3d 904, 917 (9th Cir. 2010) (overturning equitable relief extending all generations of Bratz dolls); Sandovol v. New Line Cinema Corp., 973 F.Supp. 409, 414 (S.D.N.Y. 1997).

246. Alex Kozinski & Christopher Newman, *What's So Fair About Fair Use?*, 46 J. COPYRIGHT SOC'Y OF THE U.S.A. 513, 526 (1999). Judge Kozinski argues for dispensing with both the fair use defense and injunctive relief in most cases involving a derivative work in favor of a scheme whereby the copyright holder recovers "compensation for the value arising from their work, but the derivative users are allowed to profit from the value they add." *Id.* His proposal still included a remnant of a fair use defense excluding from recovery "those damages attributable to critical evaluation of the copyright work." *Id.*

247. 17 U.S.C. section 103(a) (1976); Gracen v. Bradford Exchange, 698 F.2d 300, 302

painstakingly transforms a work to a different media with the intent of creating a perfect facsimile—creating a perfect digital rendering of an automobile for example—may not be a derivative work within the meaning of the Copyright Act if his laborious effort lacks the creative necessary for copyright protection.[248] Thus, a new work is a derivative work only if its use of a pre-existing work would be sufficient to make a *prima facie* case for infringement, ignoring for this purpose the author's right to use the prior work based on consent, a fair use or because the work is in the public domain.[249] Therefore, other than pure facsimiles, all other infringing works are derivative works within the meaning of the Copyright Act.

This conclusion is important. While the author of a derivative work arguably has established some level of new expression of a creative idea,[250] the right to make and control derivative works is one of the bundle of rights held by a copyright owner. If the owner of an indefensibly infringing work is afforded copyright protection for those portions of the infringing work that are his own original expressions of a creative idea, the right of the infringed copyright

---

(7th Cir. 1983) ("As with any painting there was an admixture of the painter's creativity — how much, we shall consider later — but that it is a painting of Judy Garland as she appears in photographs from the movie (such as the photograph reproduced at the end of this opinion as Figure 2), and is therefore a derivative work, is beyond question."); Eden Toys, Inc. v. Florelee Undergarment Co., Inc., 697 F.2d 27, 34 (2d Cir. 1982) ("A work which makes non-trivial contributions to an existing one may be copyrighted as a derivative work and yet, because it retains the "same aesthetic appeal" as the original work, render the holder liable for infringement of the original copyright if the derivative work were to be published without permission from the owner of the original copyright.").

248. Meshwerks, Inc. v. Toyota Motor Sales U.S.A., 528 F.3d 1258, 1268 (10th Cir. 2008). In affirming the district court grant of summary judgment that computer assisted 3D models of a Toyota car were not sufficiently original to warrant copyright protection, the Tenth Circuit recognized that "[i]f an artist affirmatively sets out to be unoriginal—to make a copy of someone else's creation, rather than to create an original work—it is far more likely that the resultant product will, in fact, be unoriginal." *Id.* citing RussVerSteeg, *Intent, Originality, Creativity and Joint Authorship*, 68 BROOK. L.REV. 123, 133 (2002); ATC Distr. Group, Inc. v. Whatever It Takes Transmissions & Parts, Inc., 402 F.3d 700, 712 (6th Cir. 2005) ("The illustrations were intended to be as accurate as possible in reproducing the parts shown in the photographs on which they were based, a form of slavish copying that is antithesis of originality.").

249. Litchfield v. Spielberg, 736 F.2d 1352, 1357 (9th Cir. 1984) ("We have stated that '[a] work will be considered a derivative work *only if it would be considered an infringing work if* the material which it has derived from a prior work had been taken without the consent of a copyright proprietor of such prior work.'") (quoting United States v. Taxe, 540 F.2d 961, 965 n. 2 (9th Cir. 1976).

250. While courts typically do not involve themselves in gauging the originality of a work seeking copyright protection, "a derivative work must be substantially different from the underlying work to be copyrightable." Gracen v. Bradford Exchange, 698 F.2d 300, 305 (7th Cir. 1983) (not sufficient originality in authorized painting of Dorothy from the *Wizard of Oz* to support its independent copyright ability).

holder to make and control derivative works held by the infringed copyright holder is compromised to some extent. Some commentators argue that the creators of the infringing derivative work can still maintain an infringement claim against a third party for infringement of their original expression so long as pre-existing work does not pervade the derivative work.[251] The House Report accompanying the 1976 Copyright Act does indicate that the creator of a derivative work must possess the consent of the underlying copyright holder in order to obtain protection for those portions of the work that do not employ the prior work. That report adds, however, that "a copyright could be obtained as long as the use of the preexisting work was not unlawful."[252] Where the use of the protected elements of a pre-existing work was unlawful—*i.e.* done without consent and absent a fair use defense—courts have ruled that no valid copyright existed in any portion of a derivative work.[253] While some debate remains as to the ability of the creator of a derivative work to enforce against third parties his unique original expression of creative ideas not intertwined with the unauthorized pre-existing work, the better course, as discussed below, might be to require the creator to demonstrate that his unauthorized use of the work in the context of the derivative work as a whole was fair and thus not unlawful. Moreover, even if the author of the derivative work is unable to prove his use of a pre-existing work was fair, rather than permit him to copyright the derivative work as a whole, the better course is to allow an author to protect his new expression of a creative idea to the extent he can excise his expression from the underlying work.[254] For example, the

---

251. Nimmer § 3.06 at 3-34.31 (citing cases citing treatise).

252. The House Report on the Copyright Act of 1976, at p. 57-58, reprinted in Nimmer at App. 4-23.

253. Pickett v. Prince, 207 F.3d 402, 405-06 (7th Cir. 2000), ("Concentrating the right to make derivative works in the owner of the original work prevents what might otherwise be an endless series of infringement suits posing insoluble difficulties of proof."); Schrock v. Learning Curve Int'l, 531 F.Supp. 2d 990, 996 (N.D. Ill 2008) ("Because Section 106(2) grant a copyright owner the exclusive right to prepare (or to authorize) derivative works as part of its burden of rights, a third party seeking to copyright a derivative work must have the permission of the copyright holder of the underlying work.").

254. Judge Posner in Ty, Inc. v. Publications Int'l Ltd., 292 F.3d 512 (7th Cir. 2002) appears to take a different tact when discussing the derivative nature of the *Beanie Babies Collector's Guide*, which included photographs of copyrighted Beanie Babies. While recognizing that the photographs of the Beanie Babies in the guide were derivate works, Judge Posner concludes that the text is not a derivative work. *Id.* Curiously, Judge Posner seems to believe that single work can have both derivative and non-derivative components. Judge Posner concludes that the copyright holder's provable damages should be limited to the profits from the photographs (*i.e.* the portion of the guide that is derivative) and not the profits from the work as a whole. *Id.* at 524. While an apportionment of profits for an infringing derivative work makes some sense, the mental gymnastics required to conclude that a single work can be both

author of a catalogue of works by a famous visual artist could obtain a copyright on his commentary about the artists, which can easily be separated from the images, but probably could not obtain a copyright on his photographs of the artist's work.

G. A Work Derived from Another's Expression Transforms the Original, So All Transformative Works Are Derivate Works

It is interesting that Section 103 qualifies new works that meet the definition of a derivative work as those that "recast, transform[] or adapt[]" one or more pre-existing works in order to distinguish them from simple facsimiles.[255] Since, with the exception of facsimiles, all infringing works are derivative work, are all works that transform the original expressed creative ideas of a pre-existing work *prima facie* infringing? Logic dictates that the answer to that question is yes. Assuming that use of the underlying work is sufficient to meet the requirements of a derivative work, then such transformation of the pre-existing work is infringing and unlawful absent consent or some other basis for a lawful use. Thus, the mere fact that a new work is transformative does not mean that the transformative work is not infringing. Obviously, not all transformative works are derivative works. For example, the story of an otherwise unassuming man who hides his identity as a crime fighter with super human capabilities is an idea that has been expressed as Superman, Spiderman and the Green Hornet and can be transformed into limitless other characters that are not derivative within the meaning of the Copyright Act so long as such characters do not impinge on the original expression of known superhero.[256] This same notion underscores the issue in cases involving computer software, whether what was taken was creative expression or simply a process, function or structure. This returns to the basic premise that copyright protects how you teach, but not what you teach.

---

derivative and not derivative seems unnecessary.

255. Judge Kozinski of the Ninth Circuit has noted that Copyright Act's definition of a derivative work is "hopelessly overbroad . . . for '[e]very book in literature, science and art borrows and uses much which was well known and used before." Micro Star v. Formgen Inc., 154 F.3d 1107, 1110 (9th Cir. 1998) (quoting Emerson v. Davies, 8 F. Cas. 615, 619 (C.C.D. Mass. 1845)).

256. In fact, the owners of Superman claimed copyright infringement against the network television series "The Greatest American Hero," in which a high school teacher gains superpowers to fight evil when he receives a costume from space aliens. Warner Bros, Inc. v. American Broadcasting Companies, Inc., 720 F.2d 231, 243 (2d Cir. 1983) ("The overall perception of the way Hinkley looks and acts marks him as a different non-infringing character who simply has some of the superhuman traits popularized by the Supreman character and now widely shared within the superhero genre.").

Some courts take a different tack, arguing that a second author can so transform an original expression that it ceases to be a derivative work because the expression in the new work ceases to be substantially similar to the original work.[257] This logic seems to jumble distinct concepts. To the extent that a second author copies the protectable expression of the original author, a *prima facie* case of infringement has been made. As discussed above, the substantial similarity factor, when combined with evidence of access, serves as circumstantial evidence of copying. If the second author's original expression of a creative idea is so different that it is not substantially similar to an underlying expression in any material respect, it cannot be said that the second author copied any original expression of the underlying author. Rather than there being a continuum whereby a work derived from a protectable expression can be so transformed that it ceased to be a copy, it is better to attach transformation to a continuum running from how something is taught to what is taught or from the expression of an idea, function or process and the idea, function or process itself. As similarities between expressions diverge what is being transformed moves from a protectable expression towards an unprotected idea, function or process. To the extent that the original author proves a copying of his expression, the burden shifts to the second author to prove his use fair. There is no independent defense that the original author failed to make a *prima facie* case of infringement because the second author so transformed the protected expression that it ceased to be substantially similar.

This idea of a continuum comes from Learned Hand in a quote referenced above and more fully stated here.

> Upon any work … a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended.[258]

Simply, at some level of abstraction, *all* works share an identity, as one enterprising lawyer demonstrated by outlining the similarities in plot, sequence and events between "The Wizard of Oz" and "Star

257. Well-Made Toy Mfg. Corp. v. Goffa Int'l Corp., 354 F.3d 112, 118 (2d Cir. 2003) ("Because the allegedly derivative work . . . sufficiently transformed the expression of the original work . . . such that the two works ceased to be substantially similar, 'the secondary work is not a derivative work and, for that matter, does not infringe the copyright of the original work.'").

258. Nichols v. Universal Pictures Corporation, 45 F.2d 119, 121 (2d Cir. 1930).

Wars."[259] This notion of a continuum between the idea and its expression well recognized in works of pure aesthetics should be carried over to computer software so that court recognize a continuum between the structure, process and function performed by the program and the expression that achieves that particular structure, process or function.

The issue confronted by Judge Hand in *Nichols* was not one of copyrightability but of whether the identity between the two works went beyond an identity of ideas on a continuum to reach an identity of how the idea were expressed.[260] For our purposes, the similarity continuum would be between unprotected function, process, or structure and how the function, process, or structure is expressed. As above, we must be precise again here. The relevant similarity of function, process and structure for purposes for the protection of a computer program as defined by the Copyright Act is from the prospective of the author of the program—the structure, function or process of the code itself—and not structure, function or process experienced by the user of the program. Again the distinction between *Lotus* and *Oracle*.

Judge Hand's comment about the difficult task courts face in determining when a protected expression ends and an unprotected idea begins applies when the alleged infringing work has not simply copied or plagiarized text, but allegedly has copied themes, plot or structure of a creative work.[261] This notion that copyright protects more than simply the copying of text is clear from suggestion that courts not be swayed by the differences between two works[262] or focus too much on isolated similarities[263] but should compare the

259. Shaw v. Lindheim, 919 F.2d 1353, 1363 (9th Cir. 1990).

260. Patry, *supra* note 191, at 212 ("The issue in *Nichols* was not copyrightability, but rather infringement where the defendant did not copy the text, but allegedly, characters and plot—integral parts of a unified whole."); Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 5 (1995) ("Copying violative of the statute includes not only the literal taking of the words or expression of another, but also what is called "non-literal infringement" -the taking of the essence of the author's expression without using the author's actual words. Were copyright protection limited to literal infringement, as Judge Hand has noted, a plagiarist could 'escape by immaterial variations.'").

261. Lemley, *supra* note 260.

262. Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 56 (2d Cir. 1936) (Hand, J.) ("no plagiarist can excuse the wrong by showing how much of his work he did not pirate.").

263. Comparison lists of literal similarities "between the two works are "inherently subjective and unreliable," particularly where the list contains random similarities, and many such similarities could be found in very dissimilar works." Herzog v. Castle Rock Entertainment, 193 F.3d 1241, 1257 (11th Cir. 1999) (quoting Beal v. Paramount Pictures Corp., 20 F.3d 454 460 (11th Cir. 1994); Williams v. Crichton, 84 F.3d 581, 599 (2d Cir 1996) ("[S]uch lists are 'inherently subjective and unreliable," particularly where "the list emphasizes random similarities scattered throughout the works.') (quoting" Litchfield v. Spielberg, 736 F.2d

"total concept and feel" of the two works.[264] This non-literal comparison of the two works is referred to in some jurisdictions for works of aesthetics as the "extrinsic and "intrinsic" test for substantial similarity. "The 'extrinsic test' is an objective comparison of specific expressive elements. '[T]he test focuses on articulable similarities between [all objective manifestations of expression, including] the plot, themes, dialogue, mood, setting, pace, characters, and sequence of events in two works.'"[265] "The 'intrinsic test' is a subjective comparison that focuses on "whether the ordinary, reasonable audience" would find the works substantially similar in the 'total concept and feel of the works.'"[266] Commentators and courts have questioned whether the copyright protection of computer programs should similarly reach beyond copying of the source code itself and reach similar structural—or non-literal—elements of the code.[267]

The Federal Circuit confronted this issue in *Oracle.* The court first acknowledged a confusing use of "literal," noting "[t]he distinction between literal and non-literal aspects of a computer

---

1352, 1356 (9th Cir. 1984).

264. Boisson v. Banian, Ltd*.,* 273 F.3d 262, 272-73 (2d Cir. 2001) (finding infringement on the basis of protectable and similar combinations of letters, colors and patterns in two alphabet rugs — in sum, on the basis of the "enormous amount of sameness" between the two designs).

265. Cavalier v. Random House, Inc., 297 F.3d 815, 822 (9th Cir. 2002) (quoting Kouf v. Walt Disney Pictures & Television, 16 F.3d 1042, 1045 (9th Cir. 1994) (quotation marks and citation omitted) ("the extrinsic test, now encompassing all objective manifestations of expression, no longer fits that description."); Copeland v. Bieber, 789 F.3d 484, 489 (4th Cir. 2015) ("The 'extrinsic inquiry is an objective one,' looking to specific and 'external criteria' of substantial similarity between the original elements (and only the original elements) of a protected work and an alleged copy." (quoting Dawson v. Hinshaw Music, Inc., 905 F.2d 731, 732-33 (4th Cir. 1990)).

266. Cavalier v. Random House, Inc., 297 F.3d 815, 822 (9th Cir. 2002) (quoting Kouf, 16 F.3d at 1045 (quotation marks and citation omitted); Yurman Design, Inc. v. PAJ, Inc., 262 F.3d 101, 111 (2d Cir. 2001) ("The fact-finder must examine the works for their total concept and feel.").

267. *Comput. Associates*, 982 F.2d at 712 ("[It] may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes."); Jonathan Ambrose, *Oracle America Inc. v. Google, Inc: The Only Nonliteral Aspect of JAVA API's Protected Under copyright law are the Ones Nobody Wants to Copy*, 14 N.C. J.L. & TECH. ON. 1, 13-14 (2012) ("In reality, novels enjoy a much greater scope of copyright protection for these nonliteral story elements. The comparison between these two works, one artistic and one functional, illustrates the apparent absurdity of applying copyright protection to computer programs"); Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 193 UNIV. OF DAYTON L. REV. 975, 987 (1994) ("No court, however, has quite come to grips with the fundamental question of why any nonliteral program aspect that somehow survives the filtering procedure can be deemed "expressive." Every aspect of a program, literal and nonliteral, is intended by its author to serve the functional purpose of causing the program to perform in some optimal manner in relation to the constraints set by the environment in which it is to be used (including the nature of its intended or expected users).").

program [a]s separate from the distinction between literal and non-literal copying. 'Literal' copying is verbatim copying of original expression. 'Nonliteral' copying is 'paraphrased or loosely paraphrased rather than word for word.'"[268] Google conceded it "literally copied the declaring code verbatim," which Oracle analogized to copying "'the chapter titles and topic sentences represent the structure of a novel.'"[269] By copying the declaring code and using it to trigger new implementing code that Google wrote, Oracle argued that Google infringed protected non-literal elements of its JAVA SDK because "when Google copied the declaring code in these packages 'it also copied the "sequence and organization" of the packages (*i.e.,* the three-dimensional structure with all the chutes and ladders)'" employed by the packages.[270] Significantly, even the district court, the decision of which the Federal Circuit reversed, recognized that structure, sequence, and organization ("SSO") of the JAVA API packages were creative and original.[271] The district court held, however, that the SSO was a "'method of operation' the '[d]uplication of [which was] necessary for interoperability.'"[272] Thus, there was no dispute between either the district court or the Federal Circuit that the infringement analysis for a computer program is akin to that applied by courts for aesthetic works and considers both the text itself—literal infringement—as well as the program's structure, sequence and organization—its non-literal elements. In this case, the district court and the Federal Circuit disagreed as to whether the portion of the SSO copied was copyrightable.

In this respect, the *Oracle* opinion is consistent with the practice of programming, which recognizes that what is creative about a program is more than simply the source code itself. As discussed above, among practitioners the most creative component of programming is the design of the architecture and functional specifications to implement an algorithm, believing that when properly designed the actual coding of the structure, functions and processes expressed in the design documents should be relatively straightforward. These design documents are the creative expression of the intended algorithm. In this way, coding from design documents should be almost like translating an existing text, which creative in its own respect, but does not take away from the creativity in the source

268. *Oracle*, 750 F.3d at 1356 (Fed. (quoting Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 814 (1st Cir. 1995)).

269. *Id.* (quoting Appellant Br. 27).

270. *Id.* (quoting Appellant Br. 45).

271. *Id.*

272. *Id.* (quoting 872 F.Supp.2d at 977.).

document.[273] Similarly, the design documents could be seen as a script, a play or a score, directing the action to be taken. Limiting copyright protection to the literal text, which ignoring the structure, process and function as expressed in the design document makes no sense. Any test of substantial similarity of computer programs should be akin to how copyright determines substantial similarity in other literary works or motion pictures, which is concerned with evaluating the work as a whole—including it overall mood, setting, pace, characters, and sequence of events—and not simply isolated identities.

Rather than deciding that a computer program's SSO is not creative within the meaning of the Copyright Act, the district court in *Oracle* ruled the SSO of JAVA's SDK limited Google's ability to utilize functionality of the JAVA programming language when it developed its SDK for android—*i.e.* the expression of the function had merged with the function itself.[274] A conclusion that the Federal Circuit reversed. This aspect of the Federal Circuit decision has been highlighted by the decision's critics.[275] Where the district court focused on the options available to Google to utilize the efficiencies achieved by the JAVA SDK for programming in the JAVA programming language with its SDK for android, the Federal Circuit concluded that in determining whether there are creative alternatives sufficient to distinguish expression from the intended function must be judged from the prospective of the author of the copyright work and not from the perspective of who later wants to implement the functionality.[276]

---

273. Soc'y of the Holy Transfiguration Monastery, Inc. v. Gregory, 689 F.3d 29 (1st Cir. 2012) ("The Copyright Act makes clear that translations may be original and copyrightable, despite being derivative of another product.").

274. *Oracle*, 750 F.3d at 1360 (The [district] court explained that, under the rules of Java, a programmer must use the identical "declaration or method header lines" to 'declare a method specifying the same functionality.' Because the district court found that there was only one way to write the declaring code for each of the Java packages, it concluded that 'the merger doctrine bars anyone from claiming exclusive copyright ownership' of it. Accordingly, the court held there could be 'no copyright violation in using the identical declarations.'" (citations omitted)).

275. Samuelson, *Functionality and Expression*, *supra* note 50, at 44 ("Numerous cases have taken other factors into account when assessing merger defenses, such as whether the claimed expression was the most logical and useful way to do something, whether practical considerations or functionality limited options, and whether particular designs were necessary to achieving objectives.").

276. *Oracle*, 750 F.3d at 1361 ("We further find that the district court erred in focusing its merger analysis on the options available to Google at the time of copying. It is well-established that copyrightability and the scope of protectable activity are to be evaluated at the time of creation, not at the time of infringement.")

Let's quickly consider Google's argument. In essence, because the tools included in JAVA SDK had gained widespread adoption by those who programmed in the JAVA programming language, how those tools were expressed in the JAVA SDK –*i.e.* the explicit commands used to call particular functionality—had lost any distinctive expression because, through use by programmers, they had become the most efficient way to call such commands. It is in this way commentators view *Oracle* as akin to *Lotus*. When Google decided to use the JAVA programming language as a basis for android—the JAVA programming language itself was not copyrighted[277]—Google was free, as it did, to build its own SDK to facility programmers to program in JAVA for android and to copy the function and structure of the JAVA SDK and the tools included in the program. Unlike in *Lotus*, however, where the function copied was its display to the end user of the program—Borland did not have access to the source code that implemented the displayed functionality—Google copied the actual source code and its structure to implement the functionality in its SDK. The Federal Circuit concluded that the reasons why Google copied portions of the JAVA SDK's SSO are not relevant the copyrightability of the SSO or whether its expression had merged with its function.[278] Instead, the Federal Circuit concluded, as did the Ninth Circuit in *Sega*,[279] that the reasons why expression was

277. Samuelson, *Functionality and Expression*, *supra* note 50, at 45 ("[T]he CAFC ignored the District Court's finding that there was far less of a distinction between the Java language and the API packages than Oracle acknowledged. Insofar as there is little or no difference between the Java language—which all agree is not protectable by copyright law—and the component parts of the API that Google used, this consideration weighs in favor of finding merger in Oracle."

278. *Oracle*, 750 F.3d at 1371 ("[T]he focus is on the compatibility needs and programming choices of the party claiming copyright protection — not the choices the defendant made to achieve compatibility with the plaintiff's program. Consistent with this approach, courts have recognized that, once the plaintiff creates a copyrightable work, a defendant's desire 'to achieve total compatibility... is a commercial and competitive objective which does not enter into the ... issue of whether particular ideas and expressions have merged.' (quoting *Apple Comput.*, 714 F.2d at 1253.); *compare* Samuelson, *Functionality and Expression*, *supra* note 50, at 38. ("Given that programs are utilitarian works that courts often say should enjoy only a thin scope of copyright protection, it is curious that courts have not articulated and applied a function/expression merger doctrine, as such, in software infringement cases. The reluctance to do this may stem from the misleading nature of the literary work metaphor that so often pervades copyright discourse about computer programs. This metaphor obscures the deeply functional nature of programs which have a higher quantum of utilitarian elements as compared with conventional literary works such as novels and plays.")

279. Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992) ("the record clearly establishes that disassembly of the object code in Sega's video game cartridges was necessary in order to understand the functional requirements for Genesis compatibility.")

copied are relevant to whether its copying of the expression was fair,[280] as the jury in *Oracle* ultimately held.

Let's return for a moment to the evaluation of whether the identity between computer programs relates to protectable expression or is simply an identity of unprotected functionality. This is the difficult task that Judge Hand referred to in a number of opinions concerning the determination of whereon a continuum the identity between two works ceases to be protectable expression of an idea, function, structure or process to be simply an identity between ideas, functions and structures that are being expressed differently. The analysis in *Lotus* type situation is fairly straightforward with respect to the protection of a computer program as a computer program because Borland never had access to how Lotus expressed the copied ideas, functions or processes. Irrespective of Judge Hand's conclusion that "[o]bviously, no principle can be stated as to when an imitator has gone beyond copying the 'idea,' and has borrowed its 'expression,'"[281] courts, relying on Judge Hand's lamentations, have adopted an abstraction/filtering test. [282] A test criticized by many noted jurists and commentators.[283]

Rather than viewing "what is taught" as being on a continuum with "how it is taught," consistent with Judge Hand's suggestion, the abstraction/filter tests asks courts to first separate what is copyrightable in a computer program from what is not—separate "what is taught" from "how it is taught"—and then asks the jury to evaluate only the protectable portions for substantial similarity."[284]

---

280. *Id.*

281. Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960); Mark A. Lemley*, Convergence In The Law Of Software Copyright?*, 10 BERKELEY TECH. L.J 1, 6 (1995) (Judge Learned Hand commented early and often on the arbitrariness of any such determination).

282. *Atari Games Corp*., 975 F.2d at 839 ("Judge Learned Hand devised an abstraction test to separate the idea from expression in written or dramatic works.")

283. Patry, *supra* note 191, at 201 (2003) ("Although Nimmer erroneously characterized Judge Hand's discussion in Nichols as an "abstractions test," and has successfully argued its adoption in a mutated form in computer program cases with disastrous consequences, as Judge Easterbrook wisely observed, the abstractions test is not a 'test' at all. It is a clever way to pose the difficulties that require courts to avoid either extreme of the continuum of generality.' (Nash v. CBS, Inc., 899 F.2d 1537, 1540 (7th Cir. 1990)) Similarly, Judge Newman has written that 'Judge Hand manifestly did not think of his observations as the enunciation of anything that might be called a "test." His disclaimer (for himself and everyone else) of the ability to "fix the boundary" should have been sufficient caution that no "test" capable of yielding a result was intended. (Jon O. Newman, *New Lyrics for an Old Melody: The Idea/Expression Dichotomy in the Computer Age*, 17 CARDOZO ARTS & ENT. L.J. 691, 694 (1999))".)

284. Burk, *Patenting Speech*, *supra* note 41, at 135 ("[T]he abstraction/filtration test, designed to winnow idea from expression, have been employed to separate function from expression. In this type of infringement analysis, courts first subtract from consideration

When this abstraction/filtration test is applied to software, "[t]he first step involves constructing a hierarchy of abstraction from the most abstract to the most detailed for the plaintiff's program."[285] The second step involves filtering out from the analysis various elements of the program that are beyond the scope of copyright protection."[286] This step will filter out "three kinds of unprotected elements: (1) elements of program design dictated by efficiency; (2) design choices constrained by external factors, such as the hardware and software with which the program was designed to operate, demands of the industry being served, and widely accepted programming practices; and (3) public domain elements of programs, such as commonplace programming techniques, ideas, and know-how."[287] "The third step involves comparing any remaining 'golden nuggets' of expression in the plaintiff's program with the defendant's program to determine if the defendant copied substantially amounts of expression from the plaintiffs' program. [288] Not surprising, when "properly" applied, this test results in very thin protection for computer programs due to their inherent "functional" and "utilitarian."[289] While the abstraction/filtration test comes in many flavors, most asks the court to first engage in an "analytic dissection" to separate out protectable from unprotectable elements and then to compare only protectable elements.[290] According to at least one court, "this process begins with

---

uncopyrightable elements of the works being evaluated, then they compare what remains to determine the degree of similarity between the expression contained in the works.").

285. Samuelson, *Uneasy Case*, *supra* note 59, at 1770 (2011) (described the test adopted the Federal Circuit in *Nintendo* and suggesting that it is the test "now widely adopted.")

286. *Id.*

287. *Id.*

288. *Id.*

289. *Id.*

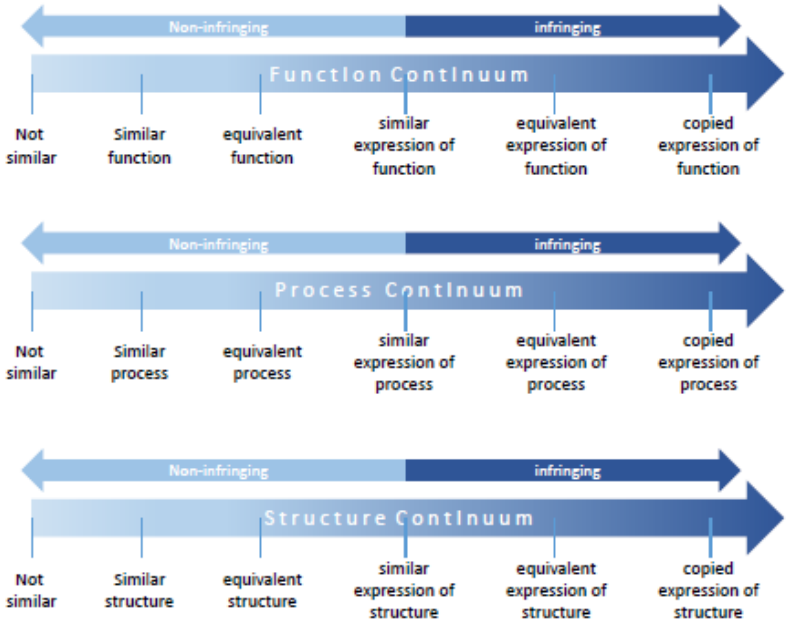
290. *Comput. Associates*, 982 F.2d at 707 ("Once the program's abstraction levels have been discovered, the substantial similarity inquiry moves from the conceptual to the concrete. Professor Nimmer suggests, and we endorse, a "successive filtering method" for separating protectable expression from non-protectable material; *See generally* 3 Nimmer § 13.03[F]. This process entails examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain and hence is nonprotectable expression."); *compare* Gates Rubber Co. v. Bando Chemical Industries, Ltd., 9 F.3d 623, 641 (10th Cir. 1993) ("We suggest that a court will often be assisted in determining the factual issue of copying if both programs are first compared in their entirety without filtering out the unprotected elements. Such a preliminary step does not obviate the ultimate need to compare just the protected elements of the copyrighted program with the alleged infringing program. However, an initial holistic comparison may reveal a pattern of copying that is not obvious when only certain components are examined.")

the code and ends with an articulation of the program's ultimate function."[291]

This test asks courts to do first what Judge Hand determined was exceedingly difficult, divining at the outset where expression ends and ideas, processes, functions and structures begin. For this very reason, this test makes no sense. A much simpler approach would be to simply follow Judge Hand's suggestion that "what is expressed" and "how it is expressed" exist on a continuum and where one begins and the other ends must be determined on a case by case bases and any algorithm to divine this demarcation would be arbitrary. Such an approach would start directly opposite from the present norm. Rather than starting with the code itself and working up, start at the highest level of abstraction to identify the similarities that exist between the accused and protected program—what are the similar ideas, function, structures and processes. Then as to each discovered similarity travel down the continuum to determine where the similarities end. If it ends before an identity of expression there is no infringement. If it passes the threshold, there is.

As mentioned, this logical test starts with the easiest step first: identify the similarities between the copyrighted program and the alleged infringing program. Second, for each separately evaluate whether their identity is limited to each: (1) performing the same function or process; (2) having the same structure; (3) or expressing the same idea. Third, if the identity between each similar feature is more than each embody the same idea, function, structure or idea, evaluate whether each achieve the same function, process or structure the same way. Fourth, if each get to the same place in similar ways, you are now in the position to determine whether they each express their ideas, functions, structures or process in such a similarly manner as to violated the Copyright Act. Finally, if copying within the meaning of the Copyright Act is found the accused infringer has the right to argue that its use is fair because (1) there are only a limited number of ways to express what is taught; (2) copying of expression was needed to achieve necessary interoperability; or (3) the public good is otherwise better served by not enforcing the Copyright Act in this context. Below of a graphic of this analysis without the fair use overlay.

291. *Comput. Associates*, 982 F.2d at 707.



H. The Problem of Fair Use.

If we accept Judge Hand's suggestion that the demarcation between expression and ideas are on a continuum, and if we also begin the fair use defense slightly before the difficult to discern point—where the expression ends and generalized idea, function, structure or process expressed begins—as was done in both *Sega* and *Oracle*, we avoid much of the controversy surrounding the protection of computer programs. Allowing for fair use justifications for copying—such as to achieve interoperability, to study function, to capitalize on familiarity, the lack of alternative ways to code, or to achieve a transformative use—seem more logical approach to evaluating a computer program than attempting to fit these justification into reasons why what was copied was not protected by copyright. With this notion the fair use overlay need not be limited to where "what is taught" intersects with "what is taught," but can extend to verbatim copying if the justification for the copying demonstrates that the public good is better served by not enforcing the copyright law is a particular instance.

The Federal Circuit correctly recognized that whether an article is creative and original—*i.e.* whether it is copyrightable—must be

evaluated at the point of its creation. A work does not lose its copyright protection because of any limits its existence may place on second comers. Neither James Bond nor Harry Potter lose their copyright protections because they now so dominate the realm of the British spy or the young wizard. One could easily argue that due to their familiarity it is much more efficient for me to write a British spy novel or a fantasy about a boy wizard by simply using James Bond or Harry Potter because I do not need to concern myself with developing their back story. In the same way, neither Windows 10 nor JAVA's SDK should lose any portion of their copyright protections simply because of their prevalence, preference, or familiarity among a desired audience. Instead, in this regard computer programs should be evaluated as any other work protected by copyright. Copyright law does not interfere with programmers' rights to use the ideas, functions, structures and processes of protected programs to make new works. Absent a legitimate justification, they do not have the right, however, to plagiarize *how* the protected work expresses that idea, function, structure or process in their new work. Such a use would result in a derivative work, which is among the rights the copyright holder retains.

The Supreme Court in *Campbell v. Acuff-Rose Music, Inc.*—the notorious case where 2 Live Crew sampled Roy Orbison's *Pretty Woman*—held that a subsequent use of a copyrighted works expression can be fair if the new use is transformative of the underlying work.[292] Some have suggested that this means that the new work's values stems not because it is a substitute of the underlying work, but results in a new work of distinct value. While *Campbell* resolved the question of whether a commercial use could be fair, it has resulted in substantial confusion about what exactly is a transformative use.[293] What must be transformed? Is the focus limited to the transformation of what was taken from the copyrighted work or should the new work be considered as a whole to determine whether a use is fair? As mentioned above, and will be elaborated on below, transformation is part of the continuum between a new work that is simply a facsimile of a copyrighted work through a derivative work and a work that merely uses the idea, functions, structures or

292. Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569 (1994).

293. *Id.* at 594 ("It was error for the Court of Appeals to conclude that the commercial nature of 2 Live Crew's parody of "Oh, Pretty Woman" rendered it presumptively unfair. No such evidentiary presumption is available to address either the first factor, the character and purpose of the use, or the fourth, market harm, in determining whether a transformative use, such as parody, is a fair one.").

processes of a work—"what is taught"—to a completely distinct work.

This continuum is a process of transformation. Importantly, however, this transformation—which results in a use being fair—can occur even when precise expression has been copied. This type of analysis explains the Ninth Circuit's decision in *Nintendo* and allowed Google to persuade a jury to find it is otherwise infringing use of the JAVA SDK fair. Freeing the copyright protection of computer programs from its existing strained copyrightability analysis is not a panacea. Existing fair use jurisprudence, however, is confused. Overlaying the issues relevant to computer programs such as interoperability, the study of functionality, and ease of use, may well help inform this area of jurisprudence and result in a more consistent approach to when a use of another protected expression is fair because to enforce the copyright laws in a particular instance would not further the public good. To understand how fair/transformative use can apply to computer programs, it is helpful to review the development of the doctrine with works of aesthetics.

3. The Fair Use Doctrine As Applied

The fair use defense is a court-made limitation on exclusivity granted authors under copyright law. While recently some have attempted to conceptualize fair use as a First Amendment limitation on copyright protection, the better explanation is that courts have recognized that mechanical application of the general provisions of the Copyright Act can fail to further the constitutional purpose underlying the Act of promoting dissemination of ideas and learning.[294] British common law recognized this tension even before Congress enacted its first copyright legislation.

> we must take care to guard against two extremes equally prejudicial; the one, that men of ability, who have employed their time for the service of the community, may not be deprived of their just merits and the reward of their ingenuity and labor; the other, that the world not be deprived of

294. Cohen, *Fair Use in the Law of Copyright*, 6 COPYRIGHT L SYMP. 43, 49 (1955). In discussing the origins of the court created fair use doctrine, Professor Cohen recognized that

> There is a strong social interest in enriching our culture and stimulating activities of a literary and artistic nature. The purpose of granting copyrights is, in the words of the Constitution, "To promote the Progress of Science and useful Arts." To deny writers fair use of copyrighted materials would have exactly the opposite effect. So, the law has been that man may make use of the work of another "for the promotion of science, and the benefit of the public"

Id. at 49 (quoting Cary v. Kearsley, 4 Esp. 168, 170 Eng. Rep. 679, 680 (N.P. 1802));

improvements, nor that the progress of the arts be retarded.[295]

Courts in the United States began reading fair use limitations into Congress's enactments as early as 1841.[296] While few questioned the need for this court created exception, courts and commentators struggled to define a standard with relevant application beyond the unique facts of a case then before a court.[297]

The problem stems from the variety of distinct uses for which the policy underlying the fair use exception is applicable, not including those outlined above for computer programs. These differing uses can be placed on a continuum from simply copying—*i.e.* time shifting held permissible in *Sony Corp. v. Universal City Studios*[298]—to uses wholly without reference to the underlying work—use of a few bars of a song in a play to add realism.[299] Unfortunately, the policy reasons for applying fair use do not fall neatly on the same continuum. Going across this spectrum, uses that present an audience more than simply a copy of the work seek fair use protection run the gambit from criticism and commentary on the work to new works whose reference to an underlying works serves purposes other than commenting on that work—Andy Warhol's Campbell Soup silk screens—to instances where a new artist uses pieces of an underlying work the same way a painter uses the color blue without any interest, intent or purpose that the audience value the reference to the underlying work.

---

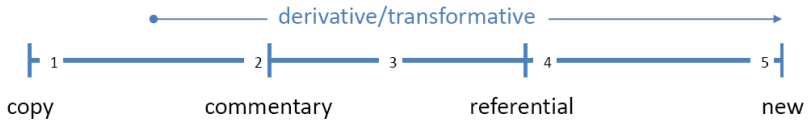
295. Sayre v. Moore, 1 East 361, 102 Eng. Rep. 139, n. 140 (K.B. 1785); Mathews Coneyer Co. v. Palmer-Bee Co., 135 F.2d 73, 85 (6th Cir. 1943) ("Like rules are applicable to the so-called fair use which may be made of copyrighted material, based upon the principle that subsequent workers in the same field are not deprived of all use thereof, as, otherwise the progress of science and the useful arts would be unduly obstructed."). Harper & Row, Publishers, Inc. v. Nation Enterprises, 471 U.S. 539, 546 (1984) ("copyright is intended to increase and not impede the harvest of knowledge").

296. *See* Patterson, *Folsom v. Marsh and its Legacy*, 5 J. INTELLECTUAL PROPERTY L. 431 (1998).

297. Sony Corp. of America v. Universal City Studios, Inc., 464 U.S. 417, 475 (1984) (The doctrine of fair use has been called, with some justification, "'the most troublesome in the whole law of copyright.'" quoting Dellar v. Samuel Goldwyn, 104 F.2d 661, 662 (2d Cir. 1939); Lawrence v. Dana, 15 F.Cas. 26, 59 (Cir. Ct. Mass. 1869) ("What constitutes a fair and bona fide abridgment in the sense of the law is, or may be, under particular circumstances one of the most difficult questions which can bell arise for judicial consideration."); Time Incorporated v. Bernard Geis Associates, 293 F.Supp. 130, 144 (S.D.N.Y. 1968) ("The doctrine is entirely equitable and is so flexible as virtually to defy definition.").

298. Sony Corp. of Am. v. Universal City Studios, Inc., 464 U.S. 417 (1984).

299. Shapiro, Berstein & Co., Inc. v. P.F. Collier & Son Co., 20 U.S.PQ. 40 (S.D.N.Y. 1934).



Across this spectrum of use and referential use, well-reasoned opinions demonstrate why based on the unique facts of a case fair use was justified for a particular use while different facts involving the same degree of referential use did not justify the application of fair use.[300]

300. Examples of differing results for similar uses as marked on the spectrum include:

1. Copying of an entire work without alteration was held to be a fair use when for the non-commercial use of time shifting (Sony Corp. v. Universal City Studios, 464 U.S. 417 (1984)) but not permissible for sharing a copyrighted work with friends (A&M Records v. Napster, 239 F.3d 1004, 1019 (2001)); Failed to bring enough new to a painting of a photograph to be protectable as a derivative work. Gracen v. Bradford Exchange 698 F.2d 200 (6th Cir. 1983). Making the cartoon character Betty Boop into a three dimensional doll was infringing. Fleischer Studios v. Freundlich, 73 F.2d 276, 278 (2d Cir. 1934).
2. Sample used in rap song could be construed as a commentary on the underlying song itself and thus a parody entitling it to fair use protection. Campbell v. Acuff-Rose Music, Inc., 510 U.S. at 580. Newsworthy excerpts of book published in magazine without "commentary, research or criticism, in part because of the need for speed if [it] was to 'make new' was not a fair use." *Harper & Row, Publishers, Inc. v. Nation Enterprises*, 471 U.S. at 543. Use of an author's unpublished letters in biography "to a large extent" made "the book worth reading" and was not a fair use. Salinger v. Random House, 811 F.2d 90, 99 (9th Cir. 1987); *compare* Wright v. Warner Books, Inc., 953 F.2d 731, 740 (9th Cir. 1991) ("The biography's use of Wright's expressive works is modest and serves either to illustrate factual points or to establish Dr. Walker's relationship with the author, not to 'enliven' her prose.").
3. Trivia game based on facts obtained from television show did not add sufficiently to the underlying, to be a fair use. Castle Rock Entertainment, Inc. v. Carol Publishing Group, Inc., 150 F.3d 132, 143 (2d Cir. 1998). Written guide to Beanie Babies held to be complementary entitled to fair use protection while the pictures of the toys themselves were not. Ty, Inc. v. Publications, Int'l Ltd, 292 F.3d 512, 519 (7th Cir. 2002). An encyclopedia of Harry Potter based on the books about the character was not a fair use. Warner Bros. Entertainment Inc. v. RDR Books, 575 F. Supp. 2d 513 (S.D.N.Y. 2008).
4. Telling of the O.J. Simpson saga in a style that mimicked Dr. Seuss but not hold the style up to ridicule was not a parody and not a fair use. Dr. Seuss Enterprises v. Penguin Books, 109 F.3d 1394, 1401 (9th Cir 1996). Changing the lyrics of "When You Wish Upon a Star" to "I Need a Jew" in an episode of the television show Family Guy was a parody and entitled to fair use protection. Bourne v. Twentieth Century Fox Film Corp., 602 F.Supp.2d 499, 511 (S.D. N. Y. 2009). Television series that brought to mind characters like superman and other superheroes not enough to be an infringement. Warner

Over time, courts have struggled to find a common thread linking the disparate instances where application of the fair use exception furthered the constitutional purpose of promoting public good from similar instances where the doctrine should have no application. Courts have (1) spoken of complementary verses superseding works,[301] (2) limited the "privilege so that it shall not be exercised to an extent that the new work inflicts substantial injury to the property which is under the legal protection of copyright;"[302] and (3) limited fair use to use of the ideas and not the expression of an underlying work.[303] Decades before Congress endorsed the fair use defense, some scholars advocated for codification to bring order to this difficult judicial doctrine, while others cautioned:

> Making the doctrine statutory might clarify it, but at the risk of compelling unjust decisions in particular cases. The important thing is to reach a just

---

Bros. Inc. v. American Broadcasting Companies, Inc., 720 F.2d 231 (2d Cir. 1983).

5. The Second Circuit rejected a fair use defense finding that an artist's sculpture developed from a photograph of puppies was not a commentary on the underlying photograph but referenced the underlying work as commentary on the commoditization of culture. Rogers v. Koons, 960 F.2d 301, 308 (2d Cir. 1992). This same artist later created another work using a fashion photograph that was targeted to the fashion genre generally rather than as a parody of the specific photographed referenced that the same court found to be a fair use. Blanch v. Koons, 467 F.3d 244, 255 (2d Cir. 2006). Well-Made Toy Mfg. v. Goffa Intern. Corp., 354 F.3d 112 ("Because the allegedly derivative work — the Huggable Lovable — sufficiently transformed the expression of the original work — the 20-inch Sweetie Mine — such that the two works ceased to be substantially similar, "the secondary work is not a derivative work and, for that matter, does not infringe the copyright of the original work," (quoting Castle Rock, 150 F.3d at 143 n. 9)).

301. Ty, Inc. v. Publications International Ltd., 292 F.3d 512, 517 (7th Cir. 2002) ("Generalizing from this example in economic terminology that has been orthodox in fair-use case law, we may say that copying is complementary to the copyrighted work (in the sense that nails are complements to hammers) is fair use, but copying that is a substitute for the copyrighted work (in the sense that nails are substitutes for pegs and screws) or derivative works from the copyrighted work . . . is not fair use."); Mura v. Columbia Broadcasting System, Inc. 245 F.Supp. 587, 590 (S.D.N.Y. 1965) (use of hand puppets on Captain Kangaroo if anything would promote sales of the hand puppets.

302. Lawrence v. Dana, 15 F.Cas. 26 (C.C. D Mass. 1869) (use of authors annotations to a new addition of a legal text in a subsequent edition of the same text was not a fair use).

303. Nichols v. Universal Pictures Corporation, 45 F.2d 119, 121 (2d Cir. 1930) (Then the question is whether the part so taken is "substantial," and therefore not a "fair use" of the copyrighted work. . . . [T]here is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended*.")* Sheldon v. Metro-Goldwyn Pictures Corporation, 81 F.2d 49, 54 (2d Cir. 1936) [T]he defendants were entitled to use, not only all that had gone before, but even the plaintiffs' contribution itself, if they drew from it only the more general patterns; that is, if they kept clear of its "expression." We must therefore state in detail those similarities which seem to us to pass the limits of "fair use."

> result in the cases that do arise. Whether the doctrine is properly applied depends, as in many areas of the law, upon the wisdom and understanding of those on the bench. [304]

Congress finally codified the fair use limitation as part of the 1976 Copyright Act.[305] In recognizing this limitation on the rights granted authors, Congress stated that the enactment intended simply "to restate the present judicial doctrine of fair use, not to change, narrow, or enlarge it."[306] With its enactment Congress expected "court [to] continue the common-law tradition of fair use adjudication." Congress' enactment states several examples and four factors it gleaned from the case law. The four factors are:

1. The purpose and character of the use, including whether such use is of a commercial nature or is not for profit purposes;
2. The nature of the copyrighted work;
3. The amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
4. The effect of the use upon the potential market for or value of the copyrighted work.

The Supreme Court in *Campbell* cautions that Congress articulated these factors and gave its examples to be "illustrative and not limiting" and to "provide only general guidance about the sorts of copying that courts and Congress most commonly had found to be fair uses."[307] The *Campbell* Court reiterated that "'[t]he fair use doctrine thus "permits [and requires] courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law was designed to foster.'"[308]

Unfortunately, before *Campbell,* mechanical application of Section 107 limited fair use to referential works of criticism and commentary on the underlying work and other non-commercial convenient uses like time-shifting, ignoring completely referential uses as a tool for new creative works or for efficiency purposes

---

304. Cohen, *supra* note 294, at 73.

305. Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 576 (1994) ("fair use remained exclusively judge-made doctrine until the passage of the 1976 Copyright Act").

306. H.R. Rep. No. 94-1476, p. 66 (1976).

307. *Campbell*, 510 U.S. at 578.

308. *Campbell*, 510 U.S. at 577 (quoting Stewart v. Abend, 495 U.S. 207, 236 (1990) (internal quotation marks and citation omitted)).

relevant to computer programming.[309] Even after *Campbell*, courts routinely draw a line at parody, which necessarily comments on the underlying work, where a satire would not be a fair use because it lacks the referential comment.[310] Some courts and commentators citing the Copyright Act purpose of fostering the dissemination of new ideas and learning, find this distinction to be arbitrary.[311] Others, however, see significance in the criticism element of parody, which complements rather than substitutes for the underlying work, and satire, which is just a humorous substitute for the original.[312] Many courts and commentators view the fair use doctrine as completely muddled.[313] Others comment that the absolute nature of the defense—free use if the defense prevails but all use enjoined if the defense fails—is the problem.[314]

One thing, however, should be clear from the above discussion, it defies explication why this court-made doctrine in equity—a determination that the application of a law in a particular instance would not further the law's purpose and therefore should not be

309. Until the Supreme Court decided *Campbell*, lower courts, relying on prior Supreme Court decisions, consistently refused to consider any commercial use to be a fair use. *Sony Corp of America v. Universal City Studios, Inc.*, 484 U.S. at 451 ("every commercial use of copyrighted material is presumptively unfair exploitation of the monopoly privilege that belongs to the owner of the copyright); *Harper & Row Publishers, Inc. v. Nation Enterprises*, 471 U.S. at 562 ("The crux of the profit/nonprofit distinction is not whether the sole motive of the use is monetary gain, but whether the user stands to profit from exploitation of the copyrighted material without paying the customary price.").

310. *Rogers v. Koons*, 960 F.2d at 308 (requiring that new work be at least a parody of the underlying work "is a necessary rule as were it otherwise there would be no real limitation on the copier's use of another's copyrighted work"); *Dr. Seuss Enterprises v. Penguin Books*, 109 F.3d at 1401 (rejecting fair use defense because mimic of style did not hold style up to ridicule).

311. Kozinski & Newman, *supra,* note 246, at 516-18 (pointing out the arbitrariness the distinction between satire and parody); at least the Second Circuit claims to have abandoned this distinction in treatment between works of parody and works of satire. (Blanch v. Koons, 467 F.3d 244, 255 (2nd Cir. 2006) ("We have applied Campbell in too many non-parody cases to require citation for the proposition that the broad principles of Campbell are not limited to cases involving parody.")).

312. Ty, Inc. v. Publication Int'l Ltd., 292 F.3d 512, 518 (7th Cir. 2002). In *Ty*, Judge Posner discussed the reason for distinguishing parody as follows:

> A parody, which is a form of criticism (good natured or otherwise), is not intended as a substitute for the work parodied. . . . A burlesque, however, is often just a humorous substitute for the original and so cuts into the demand for it . . . . Burlesques of that character, catering to the humor-loving segment of the original's market, are not fair use.

313. Dellar v. Samuel Goldwyn, Inc., 104 F.2d 661, 662 (2d Cir. 1939) ("the issue of fair use, which alone is decided, is the most troublesome in the whole law of copyright, and ought not to be resolved in cases where it may turn out to be moot").

314. Kozinski, *supra,* note 246, at 521 (arguing against the availability of injunctive relief for derivative works).

applied in the instance—has become a factual question for an ill-equipped jury to apply.

2. Why Courts Developed the Fair Use Defense and the Supreme Court's Return to the Basics

Rather than fixating on Congress's effort to capture the state of common law in 1976 and the problems with the mechanized test, it makes sense to consider again how the fair use doctrine developed, which may help clarify its application to computer programs independent of the four factor "test." In *Folsom v. Marsh*,[315] Justice Story confronted the use of more than 300 letters penned by George Washington taken from a twelve-volume collection of his writings and incorporated into a biography of the statesman entitled the *Life and Writings of George Washington*. Justice Story noted the creative step taken by the biography, "which [was] formed upon a plan different from that of [the collected works], and in which Washington is made mainly to tell the story of his own life, by inserting therein his letters and his messages, and other written documents, which such connecting lines in the narrative, as may illustrate and explain the times and circumstances and occasions of writing them."[316] Justice Story begins his opinion with the lament that "[t]his is one of those intricate and embarrassing questions, arising in the administration of civil justice, in which it is not, from the peculiar nature and character of the controversy, easy to arrive at any satisfactory conclusion."[317] In considering whether the derivative use (incorporating the copyrighted letters into a biography) was justifiable, Justice Story noted that "the defendant has selected only such materials, as suited his own limited purpose as a biographer . . . and that he has produced an exceedingly valuable book."[318] Justice Story then noted the impact to the value of the underlying work. "If so much is taken, that the value of the original is sensibly diminished, or the labors of the original author are substantially to an injurious extent appropriated by another, in point of law to constitute a piracy pro tanto."[319] In finally concluding that defendants' use was not justified, Justice Story stated "that the work of the defendants is mainly founded upon these letters, constituting more than one third of their work, and imparting to it its greatest, nay

315. Folsom v. Marsh, 9 F.Cas. 342 (Mass. Cir. 1841).
316. *Id.* at 345.
317. *Id.* at 344.
318. *Id.* at 348.
319. *Id.* at 348.

its essential value. Without those letters, in its present form the work must fall to the ground." *Id.* at 349.

The Supreme Court in *Campbell* embraced Justice Story's focus on the original creative expression brought by the second author to the derivate work, as subsequently illuminated by Judge Leval in a law review article ,[320] by noting that the

> "central purpose [of the fair use investigation] is to see . . . whether the new work merely 'supersede[s] the objects' of the original creation . . . , or instead adds something new, with a further purpose or different character, altering the first with new expression, meaning or message; it asks, in other words, whether and to what extent the new work is 'transformative. Although such transformative use is not necessary for a finding of fair use . . . the goal of copyright, to promote science and the arts, is generally furthered by the creation of transformative works. Such works thus lie at the heart of the fair use doctrine's guarantee of breathing space within the confines of copyright, and the more transformative the new work, the less will be the significance of other factors like commercialism, that may weigh against a finding of fair use.[321]

Following *Campbell's* broad language recognizing the importance of the transformative purpose made by the second author, courts and commentators cease upon this consideration in evaluating the fair use defense.[322] While continuing to acknowledge the four factors in the Copyright Act, commentators have been emboldened to question whether these factors serve the purpose intended by the judicial created doctrine, and courts continue to expand the reach of

---

320. Pierre N. Leval, *Toward a Fair Use Standard*, 103 HARV. L. REV. 1005, 1112 (1990). Judge Leval's article appears prompted in some part by his disagreement with Second Circuit's analysis in Salinger v. Random, 811 F.2d 90 (2d Cir. 1987) and New Era Publications Int'l v. Henry Holt & Co., 873 F.2d 576 (2d Cir. 1989), which gave only limited consideration to the contribution of the second author, which both involved the use of the subjects unpublished letters by biographers without consent. *Id.* at 1112-1114. Reading the first of Congress' fair use factors as direct to the second authors transformative purpose, Judge Leval concludes that "[t]he first fair use factor calls for a careful evaluation whether the particular quotation is of the transformative type that advances knowledge and the progress of the arts or whether it merely repackages, free riding on another's creation." *Id.* at 1116.

321. *Campbell*, 510 U.S. at 579 (citations and quotations omitted).

322. Prior to *Campbell*, courts routinely dispensed with fair used defenses by mechanically citing language from the Supreme Court's prior fair use decision that "every commercial use of copyrighted material is presumptively an unfair exploitation of the monopoly privilege that belongs to owner of the copyright." Sony Corp. of America v. Universal City Studios, Inc., 464 U.S. 417, 451 (1984) (holding that consumers non-commercial time shifting of copyrighted material to be a fair use). Similarly, courts seemed predisposed against finding a fair use for new works of fiction or fantasy as opposed to factual or non-fiction works. *Micro Star v. Formgen, Inc*, 154 F.3d at 1113 ("The fair use defense will be much less likely to succeed when it is applied to fiction or fantasy creations, as opposed to factual works such as telephone listings." (citing United Tel. Co. v. Johnson Publ'g Co., 855 F.2d 604, 609 (8th Cir. 1988); (Stewart v. Abend, 495 U.S. 207, 237 (1990).

the fair use principles to situations where the second author has created something truly new.

Unfortunately, courts and commentators following *Campbell* have struggled with how to reconcile the concepts of a derivative use, fair use, and transformative use—*i.e.* the point where although expression is taken it should be considered fair.[323] As discussed above, all derivative works transform an underlying work to some extent. This does not mean, however, that all derivative works are transformative works within the meaning of *Campbell.* Nothing in *Campbell* can be read to narrow a copyright holder's right to derivative works as provided in Section 106. Courts simply should not be uncomfortable considering the same evidence to determine whether a new work is a derivative and whether a work is sufficiently transformative to warrant a finding of fair use. On the other hand, a finding that a new work is a derivative work within the meaning of Section 106 is not required to conclude that the new work makes a transformative use of the underlying work within the meaning of *Campbell.* The new work can be a transformation of the idea underlying an expression, even if the expression is copied.

It is also important to remember that the *Campbell* Court did not substitute its conclusion that a new work can be sufficiently transformative to be a fair use for the four court-created factors recognized by Congress in Section 107. New uses of copyrighted material can be fair even if the new works simply copies the underlying work—*i.e.* there is no transformation of expression sufficient to consider the work to be a derivative work. Recent cases involving the limited use of copyrighted materials as part of a news story,[324] for purposes of facilitating searches on the internet,[325] or in historical works of non-fiction[326] can and should be considered fair

323. *See* Reese, *Transformative and Derivative Work Rights*, 31 COLUM. J.L. 467 (2008) ("I conclude that appellate courts do not view fair us transformativeness as connected with any transformation involved in preparing a derivative work, and that in evaluating transformativeness the courts focus more on the purpose of a defendant's use than on any alteration the defendant has made to the content of the plaintiff's work.").

324. Fitzgerald v. CBS Broadcasting, Inc., 491 F.Supp. 2d 177, 184 (D. Mass. 2007) ("News reporting is one of the categories of use expressly mentioned in the Copyright Act as particularly appropriate to the application of fair use doctrine.").

325. Perfect 10, Inc. v. Amazon.com, Inc., 487 F.3d 701, 725 (9th Cir. 2007) ("Google has provided a significant benefit to the public. Weighing this significant transformative use against the unproven use of Google's thumbnails for cell phone downloads, and considering the other fair use factors, all in light of the purpose of copyright, we conclude that Google's use of Perfect 10's thumbnails is a fair use.")

326. Wright v. Warner Books, Inc., 953 F.2d 731, 740 (9th Cir. 1991) ("The biography's use of Wright's expressive works is modest and serves either to illustrate factual points or to establish Dr. Walker's relationship with the author, not to 'enliven' her prose.").

uses irrespective of whether these uses can be considered transformative.

This analysis should be equally applicable to computer programs to judge whether the expression taken is a fair use. One could argue that the reason for the strained focus on copyrightability found in many cases addressed to computer programs stems from the confusion around fair use. Although not a point expressed in any of those opinions, the analysis in *Sega* and the jury's decision in *Oracle* indicates that fair use provides a firmer footing to evaluate the public good served by the application of copyright to a particular situation than copyrightability or the merger doctrine.

What *Campbell* brought to fair use jurisprudence was the Court's reaffirmation that consideration of the purpose and character of the new work's use of the copyrighted material does not begin and end with whether the new use was commercial or not.[327] The *Campbell* Court clarified that "[n]o 'presumption' or inference of market harm that might find support in *Sony* is applicable to a case involving something beyond mere duplication for commercial purpose."[328] Stated another way, does the commercial value of the new work created for a commercial purpose derive merely from what has been duplicated—resulting in a presumption of against fair use—or is its commercial value derived primarily from what is new in the new work—a potentially transformative work where no presumption is appropriate?[329]

Dispensing with a myopic focus on whether the new work was commercial or not, the *Campbell* Court freed fair use to reach new creative works that incorporate elements of existing works in new and different ways. In fact, the *Campbell* Court discussion of the fourth statutory fair use factor—the effect upon a potential market for value of the copyright work—narrowed "the only harm . . . that need

---

327. *Campbell*, 510 U.S. at 583-84. The *Campbell* court, however, does draw a distinction between artistic works "The use . . . of a copyrighted work to advertise a product, even in parody, will be entitled to less indulgence under the first factor of the fair use enquiry, than the sale of the parody for its own sake" *Id.* at 585.

> "The Court of Appeals . . . immediately cut short the enquiry into 2 Live Crew's fair use claim by confining its treatment of the first factor essentially to one relevant fact, the commercial nature of the use. The court then inflated the significance of this fact by applying a presumption ostensibly culled from *Sony* that 'every commercial use of copy-righted material is presumptively . . . unfair' *Sony*, 464 U.S. at 451. In giving virtually dispositive weight to the commercial nature of the parody, the Court of Appeal erred." *Id.* at 584.

328. *Campbell*, 510 U.S. at 591.

329. As will be discussed *infra,* recognizing transformative use as a consideration of the reason for the commercial value of the new work has allowed court to avoid the quagmire of what is and is not a commercial use when considered a right of publicity claim.

concern us . . . is the harm of market substitution."[330] Irrelevant is harm resulting from a derivative work "that adds something new, with a further purpose or different character, altering the first with new expression, meaning or message."[331] Seizing on this language, some argue post-*Campbell* that the complexities of using the language of transformation and substitution can be meliorated by utilizing tort lexicon of cognizable harm.[332] Stated simply, in deciding whether a new use is fair, a court can consider whether the underlying copyright holder has suffered a copyright injury[333] sufficiently definite to be compensable under ordinary tort law principles.[334]

---

330. *Campbell*, 510 U.S. at 593; *see Mattel Inc. v. Walking Mountain Productions*, 353 F.3d at 806. In *Walking Mountain*, the owner of Barbie sued a photographer who incorporated Barbie Dolls in his photographic works in less than a flattering manner, *see e.g.* Barbie in a blender. In affirming the grant of summary judgment for the artist based on a finding of fair use, the Ninth Circuit recognized that the new "work could only reasonably substitute for a work in the market for adult-oriented artistic photographs of Barbie. We think it is safe to assume that Mattel would not enter such a market or license other to do so." *Walking Mountain Productions*, 353 F.3d at 806. The court continued, "the public benefit in allowing artistic creativity and social criticism to flourish is great. The fair use exception recognizes this important limitation on the rights of the owners of copyrights. No doubt, Mattel would be less likely to grant a license to an artist that intends to create art that criticizes and reflects negatively on Barbie's image. It is not in the public interest to allow Mattel complete control over the kings of artistic works that use Barbie as a reference for criticism and comment." *Id.* Curiously, the *Walking Mountain* Court does not discuss transformative use. Instead, the Court focused on parody even though its discussion of parody seems to reach beyond comment on Barbie to include social commentary as well. *Id* at 802.

331. *Campbell*, 510 U.S. at 579. Simply editing out objectionable elements of a film to make a "clean" version available does not add something original as contemplated by a transformative use. *Clean Flicks of Colorado, LLC v. Soderbergh*, 433 F.Supp.2d at 1241 ("The transformative nature of the use of copyrighted material requires such a contribution of originality as may be of such public benefit as would serve the underlying purpose of providing copyright protection").

332. Thomas F. Cotter, *Transformative Use and Cognizable Harm*, 12 VANDERBILT J. OF ENTM'T. & TECH. L. 701, 701 (Professor Cotter "argues that much of the current emphasis on transformative use is misguided and that court instead should focus the fair use inquiry on whether the defendant's unauthorized use threatens *cognizable harm* to the copyright owner's interest in exploiting her work.") (emphasis in original).

333. Not all harm resulting from another use of a copyrighted material is an injury compensable under copyright law. For example while a well-publicized negative review of a book will likely have a greater economic impact on an author than the decision of a famous filmmaker to create a dramatic work from an obscure novel, only the latter gives right to an injury compensable under copyright law if the film is made without a license. *See Campbell,* 510 U.S. at 591-92 ("a lethal parody, like a scathing theater review, kills demand for the original, it does not produce a harm cognizable under the Copyright Act.")

334. Looking at fair use in this context the battle lines in digital sampling become rather clear. The owner of the underlying copyright claims harm because the defendant could have obtained a license and the failure to obtain a license is a cognizable harm to the copyright holder. The creator of the new work might argue that the terms of any license are so economically prohibitive that the underlying copyright holder is actually blocking any substitutional work. As the Supreme Court noted in *Campbell* the underlying rights holder to *Pretty Woman* may not have been willing to license 2 Live Crew's sample due to objections to

The *Campbell* Court's active reconsideration of fair use and lower court's continued struggle to grasp competing jurisprudence in this area reconfirms the difficult policy balance that must be struck in deciding whether a particular use should be protected as fair. Trial courts, whether intentionally or subconsciously, routinely treat the doctrine as equitable summarily deciding the issue while balancing facts, and reviewing courts continue to rebalance the facts *de novo* in their evaluation as to whether the policies underlying fair use when applied to the particular facts.[335] While some argue that courts are improperly intruding on what should be a matter for the jury to decide, where, as here, courts themselves have a difficult time wrestling with this issue,[336] it seems time that this duck is finally called a duck, and fair use be treated as the equitable defense that it is.[337] The move towards this inevitability is indicated by one court's conclusion that "every court to address the issue whether a defendant's work qualifies as a parody"—the purpose and character

---

the new song. *Id.* Courts have cited a rights holder's likely unwillingness to license the new work as a reason to find a new transformative use fair. A distinction between a rights holder's willingness to license the new use, however, seems like an arbitrary demarcation for determining whether a use is fair.

335. *Harper & Row, Publishing, Inc. v. Nation Enterprises*, 471 U.S. at 560 ("Where the district court has found facts sufficient to evaluate each of the statutory factors, an appellate court 'need not remand for further fact finding . . . [but] may conclude as a matter of law that [the challenged use] do[es] not qualify as a fair use of the copyrighted work.'" (quoting Pacific & Southern Co. v. Duncan, 744 F.2d 1490, 1495 (11th Cir. 1984))).

336. *See* Sony BMG Music Entertainment v. Tenebaum, 672 F.Supp.2d 217 (D. Mass. 2009). "Before addressing the merits of the fair use defense…[the *Tenebaum* Court] inquired whether fair use was historically an 'equitable defense' as a number of other courts have suggested. Since two leading copyright historians suggested that the equitable label may be a misnomer, and since neither party pressed the point, the Court will assume that fair use is a jury question, and leave the equitable origins of this defense for anther court to answer." *Id.* at 223.

337. In one of the earliest articulations of the fair use defense, the court indicated that it would leave it for the jury to decide whether what was "'taken used fairly.'" *Cary v. Kearsley*, 4 Espinasse 168 (1802). *See* Bradshaw, *Fair Dealing" as a Defense to Copyright Infringement in UK Law: A Historical Excursion from 1802 to the Clockwork Orange Case 1993*, Irrespective of Lord Ellenborough's willingness to let a jury decide whether what was taken by a subsequent author was fair, Congress in codifying fair use expressly recognized the defense's equitable nature. The House Report expressly states

> "Although the courts have considered and ruled upon the fair use doctrine over and over again, no real definition of the concept has ever emerged. Indeed, since the doctrine is an equitable rule of reason, no generally applicable definition is possible, and each case raising the question must be decided on its own facts."

Sony Corp. of America, 464 U.S. at 448 n. 30 (1984) (quoting House Report). Since most agree that patent and copyright misuse are equitable defenses one wonders why the same reasoning does not extend to fair use. *See* Morton Salt Co. v. G.S. Suppiger, 314 U.S. 488, 490-92 (1942), (recognizing patent misuse to be an equitable defense); Lasercomb America, Inc. v. Reynolds, 911 F.2d 970, 978 (1990) (recognizing copyright misuse as an equitable defense).

of use fair use factor—"has treated this question as one of law to be decided by the court."[338]

III. CONCLUSION

This article has covered a lot of ground and many of its conclusions are likely far from uncontroversial. I suggest that a computer program can be protected under the Copyright Act in more than one way. It can be protected in the manner that users interact with it most often—as an audiovisual work. It can be separately protected as a literary work, the manner in which Congress has expressly identified under the Copyright Act. These two forms of protection should be seen as distinct, in the same way we distinguish between the protection of a sound recording and its written composition. The meaning of "function" for our use of a computer program as an audiovisual work differs from the meaning of "function" as it relates to the protection of the computer program as text. This is the distinction between *Lotus* and *Oracle* discussed above. Additionally, it raises the importance of access to a computer program as text, which is occulted when source code must be compiled to run on a computer.

The Copyright Act's treatment of programming as literary is consistent with the actual practice of programming. While many have questioned the wisdom of this choice, suggesting that programming is more akin to industrial design, the practice of programming simply does not lend itself to the objective standards common to other mechanized functions, processes or structures. The act of programming is much more like that of writing literature, and it is appropriate to recognize the creativity expressed in the architecture and design documents that instruct the code ultimately written—as we do with the structure of a novel or film—and not to elevate the source code itself, which would be inconsistent with what programmer see as the value of what they do. Similarly, viewing programming as literature easily distinguish the craft from the implementation of ideas functions or process that are protected by patent law.

Efforts to limit copyright protection of a program's expression to allow for interoperability and to ease use—relying on the "merger doctrine" and expanded notions of functionality—seem tortured. The Federal Circuit in *Oracle* correctly recognized that what is protectable about a computer program must be decided from the perspective of

338. *Mattel, Inc. v. Walking Mountain Productions*, 353 F.3d at 801 (citing *Campbell*, 510 U.S. at 582-83; *Leibovitz v. Paramount Pictures Corp.*, 137 F.3d F.3d 109, 114-15 (2d Cir. 1998); *Dr. Seuss*, 109 F.3d at 1400-01).

the author at the time of creation. Nothing supports that a copyrighted work loses some of its protections as the work becomes more popular. Instead, the rights of others to use protected expression should be decided based on the doctrine of fair use.

The article suggests that the abstraction/filtration test currently employed to dissect protected expression from unprotected ideas, structures function or process is impossible. Asking court to start with a program's source code and work back to differentiate "what is taught" from "how it is taught" is asking too much from judges not steeped in programming. Rather than first separating "what is taught" from "how it is taught" so that the jury only focuses on the expressive teaching, it seems to make much more sense to first identify similarities in what is taught and then see if the copyright holder can prove that what is shared extends to reach the manner of teaching. If the copyright holder succeeds in proving that more was taken than "what was taught," the defendant is then free to demonstrate that the new work is transformative of what was taken, such that the policies underlying the Copyright Act are not further by its application to the unique case. Whether the purpose of a law is further by its application to a particular case, however, has never a question for a jury—unfamiliar with the underlying purpose of any law—to decide. So it makes no sense for jury to be asked to decide whether an infringing use of a protected work should be excused in order to further the underlying purposes of the Copyright Act, especially when that is at issue is a computer program.

While some argue that copyright hinders rather than furthers the public good with respect to computer programs, it is reasonable to conclude that the ambiguity of the Act's application to computer programs rather than the Act itself is what stifles innovation. Giving an author the ability to protect their creative expression does not mean they must use the tool. Similarly, as is the case with FOSS's co-opted copyleft platform, the availability of copyright protection can be used not only for monetary ends, but to support a political agenda. It is the lack of comfort that copyright will protect an author's source code that likely causes authors to hide their code from the public. If enforcement was clear and readily available, source code could be more widely disseminated because authors could be confident that the taking of their expression in a manner inconsistent with terms of use—whether those terms or monetary or otherwise—could be stopped.

Obviously, our copyright laws are not objectively correct or objectively economically optimized, nor could they ever be.

Intellectual property laws like all laws are simply rules groups of people agree to either voluntarily or coercively. Rather than right or wrong, they are simply political. Their accuracy should be compared with the rules governing sports or games rather than statistics. We chose to live by rules not for their correctness, but because we do not trust our neighbor to honor the general principles of liberty; we each have the freedom to act to the extent our actions do not interfere with the rights of others in a manner that they would not consent to as reasonable. This is not to say that laws, and in particular, intellectual property laws, do not have social and economic ramification. Laws and the absence of laws, however, both have ramifications. Any law will have winners and losers, just as the absence of laws will have the same, but line drawing is simply political.

Many speak about abolishing intellectual property laws because they advantage some over others and they can be said to have both beneficial and deleterious effects on the "common good." The Founding Fathers faced this issue—what to do with ownership of property and in particular what to do with ownership of intellectual property. The Founding Fathers, while vaguely, rightly or wrongly concluded that the federal government may regulate in this area in order to promote science and the useful arts. The Founding Fathers concluded the public good is furthered by promoting the development of science and the useful arts within the country.

It is true that the owners of these rights granted by Congress can use them to extract economic benefit—a motivation for many. More recently, however, those who could benefit from such rights have pursued such rights because they grant benefits that are non-economic as well. The possessor of such rights can influence how their rights are used by others even if the benefit they seek is not money. They extract other concessions in exchange for use—such as requiring open and public dissemination of any use (*e.g.*, requiring open source of any software code).

For those who see intellectual property law as retarding rather than furthering the public good, their view may be too narrow. Just as changes to any intellectual property law changes the balance of winners and losers, nothing supports that the abolition of intellectual property laws would result in an egalitarian utopia. And this is not simply saying that content creators deserve compensation for their creations.

This is especially true today with technology. If we abolished all copyright protection on music and allowed endless versions of Napster to exist, technology today tells us that this does not mean we

will all have free access to all of the music that we want. Streaming now means that access does not require transfer of a file. Just think of the current trend of exclusive music releases. Technology itself imposes the barriers to access.

This brings us to software. The absolutism of many with respect to how software should be treated under the law misses the nuances. The correct legal regime under which software can or should be regulated or whether it should be regulated at all is a political decision—there is no economically or socially correct answer. Whatever Congress decides to do with respect to its authority in this area will have winners and losers.

Similarly, many who write and think in the area of software and law incorrectly take a rather myopic view of all of the various manifestations that fall within the term software. In this regard, software is something like painting. Painting includes the utilitarian act of painting the wall of a house, to the painting of a sign, to a work of fine art. Are the all the same? Software is like architecture. No one would say that architecture is simply a completed structure, but includes sketches drawing, blueprints, models, construction, utility and creativity. We all recognize that all software must be written and for this reason falls under the rubric of literature in the Copyright Act. But writing is not myopic. Writing includes news copy, a phone book, or a great novel. We do not treat all writings the same nor should we treat all software or all aspects of software the same way.

The law has looked at software as utilitarian. Historically, software and hardware entered commerce together with specifically designed machines having specifically designed instructions for their function. Software, however, is now divorced from hardware; today, the two are independent. Moreover, hardware has ceded more and more ground to software. Today we have an ever more generic piece of hardware capable of ever more unique and difference activities because of the differing instructions it is now capable of following. Moreover, those involved in coding view code today as much more than simply function. Today in the art world, code is simple a tool of expression no different than paint and a paintbrush or an electric guitar. This brings us back to Autechre. Autechre does not simply playing virtual instruments on a laptop computer, but instead create code that generates music on its own or in collaboration with other code or a person. Code in this instance is no more a functional utility than a guitar solo by Jimi Hendrix is a functional utility because all he was doing is manipulating a guitar. Many will argue that what Hendrix was doing was purely aesthetic, had no utility, and therefore

properly falls within the realm of copyright. Anyone who suggests that should listen to a song by Autechre.