Software Copyright and Innovation after *Oracle v. Google*

by Sean Hogle

Published in Winter 2015 *New Matter* (California State Bar Intellectual Property Section Quarterly Journal).

 Sean (linkedin.com/in/epiclaw) practices in the areas of technology, intellectual property, and commercial law, and is the CEO and founder of Redline (redline.net), an exclusive collaboration environment for lawyers. Sean spent part of his career as Assistant General Counsel at Sun Microsystems, responsible for Java platform licensing.

In 1995, Sun Microsystems released the Java platform and programming language.[1] The Java language, released for free use by all, is one of the most popular programming languages in the world today, taught in university computer science departments worldwide.[2] Developers have published thousands of Java applications for servers, personal computers, and smartphones.

[1] James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley, *The Java Language Specification: Java SE 8 Edition* (2015).

[2] "All agree that Google was and remains free to use the Java language itself." Oracle America, Inc. v. Google Inc. ("Oracle I"), 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012) (*passim*). Java is both an application execution platform and a programming language to write applications that will run on this platform. Java is "designed for the distributed nature of the Internet." TechTarget Search SOA, *Java Definition*. Sun sought to disseminate the platform as widely as possible in order to promote the objective of "write once, run anywhere" Java application portability. *See infra* n. 39.

In 2007, Google launched the Android operating system for mobile devices, and released the source code of that system under a royalty-free and permissive open source license for free use by device manufacturers.[3] To promote partial interoperability between the Java and Android platforms, Google copied a subset of the collection and organization of names, or the application programming interfaces ("APIs"), that Sun used for labeling and classifying certain functions in the Java language and platform, and included these interfaces in Android for use by application developers.

[3] *See Welcome to the Android Open Source Project!*, Google; *Android (operating system)*, Wikipedia.

Oracle acquired Sun in 2010, and in that same year, Oracle sued Google for copyright and patent infringement in the United States Northern District of California. After a jury trial in which Oracle's patent claims were rejected, Judge Alsup ruled that the Java APIs Google copied in order to promote interoperability between the Java and Android platforms constitute an uncopyrightable method of operation under Section 102(b) of the U.S. Copyright Act ("Section 102(b)").[4]

[4] *Oracle I,* 872 F. Supp. 2d at 974.

Oracle appealed this decision to the U.S. Court of Appeals for the Federal Circuit ("CAFC" or the "Federal Circuit"). Because Oracle's case included patent claims, the CAFC had jurisdiction over the appeal, and so Oracle was able to avoid review by the U.S. Court of Appeals for the Ninth Circuit. The CAFC,

purportedly applying Ninth Circuit case law, reversed Judge Alsup, and held that the Java APIs are copyrightable.[5] The CAFC ruled that considerations of interoperability are relevant only with respect to the question of whether Google's adoption of the Java APIs was fair use, and remanded to the district court to resolve the fair use issue.

[5] Oracle America, Inc. v. Google Inc. ("Oracle II"), 750 F.3d 1339 (Fed. Cir. 2014) (*passim*).

In June of this year, the U.S. Supreme Court declined to hear an appeal from the Federal Circuit, leaving the CAFC decision intact.[6] The precedential power of the Federal Circuit's decision, however, is limited; the CAFC was constrained to follow Ninth Circuit precedent, and the Ninth Circuit can readily reject the CAFC's reasoning in future cases. The Ninth Circuit should do so at the earliest opportunity.

[6] Google, Inc. v. Oracle America, Inc., No. 14-410 (U.S. June 29, 2015).

The Federal Circuit's decision is deeply, dangerously flawed. The CAFC failed to grasp the significance of what Google copied and why Google copied it. The court labored under a misapprehension of the facts regarding the nature and significance of the Java APIs in question, and the level of interoperability between the Java and Android platforms. Further, the appellate court erroneously dismissed Google's interoperability objectives as a fair use concern, failing to realize that interoperability is the key to understanding whether APIs constitute an uncopyrightable "method of operation" under Section 102(b) of the Copyright Act.

What exactly did Google copy and why did it copy it?

Google designed software that uses the same "declaring code" that Sun used, so that Android applications, seeking to invoke a set of defined functions, could call those functions using the same labels that Sun used for Java applications. This made things significantly easier for Java application developers writing new applications for, or migrating existing Java applications to, Android.

To be precise, Google copied the following naming convention used by Sun:

```
java.package.class.method()
```

A package is a set of classes, and each class contains a set of methods, within which are sets of functions; an example of a declaration following this format is java.lang.Math.max, and an example of code that would call this function, using this declaration convention, is:

```
int a = java.lang.Math.max (2, 3);
```

This command returns the greater of two inputs (the "max"). It directs the program, as Judge Alsup explained, to "find the max method under the Math class in the java.lang package, input '2' and '3' as arguments, and then return a '3', which would then be set as the value of 'a.'"[7]

[7] *Oracle I*, 872 F. Supp. 2d at 972, 981.

Google's platform included only these declaration names that Sun used, and not the Sun/Oracle implementing code to which the functionality applied.[8] In other words, Google developed functionality from scratch, but made that functionality available to Android applications using (in part) the same declaratory convention that Sun used.

[8] *Oracle I*, 872 F. Supp. 2d at 978. Note that Google did literally copy nine lines of code in the "range.check" package and eight decompile security files (which never found its way into the Android platform). *Id.* at 983; *Oracle II*, 750 F.3d at 1351. This fact was not material to the copyrightability analysis at either the district or appellate court level.

Why would Google copy only the labels that Sun used to describe and invoke functionality that Google developed on its own? "Google believed Java application programmers would want to find the same … functionalities in the new Android system callable by the same names as used in Java."[9] As Google explained in its briefing to the CAFC, "Google supported use of the [Java programming language ("JPL")] by Android developers because the JPL was widely taught in universities and was the best environment for software development."[10]

[9] *Oracle I*, 872 F. Supp. 2d at 978 (emphasis added).

[10] Brief for Google Inc. as Appellee and Cross-Appellant at 18, Oracle America Inc. v. Google Inc., No. 13-1021, 1022 (Fed. Cir. May 23, 2013).

Section 102(b) defines the scope of copyright protection for software

By virtue of Section 102(b) of the U.S. Copyright Act, copyright does not extend to "any idea, 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."[11] Such systems and methods should be free for anyone to use unless patent law applies to protect them. Admittedly, Section 102(b) has always posed a bit of a dilemma for software, as each one of a computer program's instructions can readily be described as a "procedure, process, system or method of operation". Nevertheless, clear lines were well-drawn and well-understood in this context since at least the time of when Section 102(b) was added to the Copyright Act in 1976.

[11] 17 U.S.C. § 102(b).

Section 102(b) is a codification of the venerable U.S. Supreme Court decision of *Baker v. Selden*,[12] in which the Court ruled that the author of a book describing a new accounting system could not use copyright law to prevent others from using that system. In connection with the 1976 amendments to the Copyright Act, Congress added Section 102(b) specifically because of concerns about extending copyright protection to software, as the House and Senate report to the 1976 Copyright Act amendments reveal:

> Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer …. 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.[13]

[12] 101 U.S. 99 (1879).

[13] H.R. REP. No. 94-1476, at 56-57 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670. Congress enacted Section 102(b) "as a result of the debate over the copyrightability of computer programs," in order to prevent overprotection of computer software. *See* United States Copyright Office, *General Guide to the Copyright Act of 1976* at 3:3–3:4 (Sept. 1977). *See also* Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEX. L. REV. 1921, 1929-30 (2007).

Per the recommendation of the Congressionally-established National Commission on New Technological Uses of Copyrighted Works ("CONTU"), Congress amended the Copyright Act in 1980 to include a definition of "computer program" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." The CONTU report to Congress explained how expressly extending protection to software is reconcilable with Section 102(b):

> Section 102(b) is intended … to make clear that the expression adopted by the programmer is the copyrightable element, and that the actual processes or methods embodied in the program are not ….
> The way copyright affects games and game-playing is closely analogous: one may not adopt and republish or redistribute copyrighted game rules, *but the copyright owner has no power to prevent others from playing the game*.[14]

[14] *Oracle I*, 872 F. Supp. 2d at 986 (quoting NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, Final Report 20 (1979)) (emphasis added).

Section 102(b) has defined the scope of copyright protection of software for decades.[15] The history of software and computing since its enactment validates the efficacy of the balance that Congress struck in this regard. IBM's circa-1980s operating system powering the first mainstream generation of personal computers became the dominant computing paradigm of its time because of the open and non-proprietary nature of the APIs for this operating

system.[16] Other personal computer manufacturers were able to emulate the IBM basic input/output system (BIOS) without fear of copyright liability, paving the way for numerous PC clones.[17]

[15] Similarly, the European analogue to Section 102(b), also established for decades, is the 1991 EU Software Directive, which states, "[i]deas and principles which underlie any element of a computer program, *including those which underlie its interfaces*, are not protected by copyright …." (emphasis added). Art. 1(2), Council Directive 91/250/EEC of 14 May 1991 On the Legal Protection of Computer Programs.

[16] Corrected Brief for Computer Scientists as *Amici Curiae* Supporting Defendant-Cross Appellant at 4-25, Oracle America Inc. v. Google Inc., No. 13-1021, 1022 (Fed. Cir. May 30, 2013); Brief for Intellectual Property Law Professors as *Amici Curiae* Supporting Defendant-Cross Appellant, at 7-10, Oracle America Inc. v. Google Inc., No. 13-1021, 1022 (Fed. Cir. May 30, 2013).

[17] Corrected Brief for Computer Scientists as *Amici Curiae* Supporting Defendant-Cross Appellant at 4-25, Oracle America Inc. v. Google Inc., No. 13-1021, 1022 (Fed. Cir. May 30, 2013).

An explosion of innovation took place in a legal environment where everyone was free to implement the same APIs as utilized by IBM, Microsoft, Apple, HP, Sun and a myriad other enterprises, standards, and projects, as methods of operation clearly excluded from copyright protection by virtue of Section 102(b). A legal regime that fostered unfettered software interoperability resulted in robust competition and innovation in a bewildering array of computing platforms and applications.[18] It's no exaggeration to say that Linux, MacOS, the C programming language, cloud computing, the Internet, and even the Java platform, owe their existence in part to the historically open and non-proprietary nature of APIs.[19]

[18] *See* Jonathan Band & Masanobu Katoh, *Interfaces on Trial 2.0* (2011), at 31 ("[P]reventing interoperability invariably stifles creativity; new firms cannot introduce new products for the locked-in base, and the monopolist has little incentive to innovate—and innovation is the ultimate goal of the intellectual property system.")

[19] Corrected Brief for Computer Scientists as *Amici Curiae* Supporting Defendant-Cross Appellant at 4-25, Oracle America Inc. v. Google Inc., No. 13-1021, 1022 (Fed. Cir. May 30, 2013). *See also* Brief for Hewlett-Packard Company, Red Hat, Inc., and Yahoo! Inc. as *Amici Curiae* Supporting Petitioner at 6-14, Google Inc. v. Oracle America, Inc. No. 14-410 (U.S. Sup. Ct. Nov. 7, 2014). The Java platform itself owes its success to the non-proprietary nature of APIs. *See infra* n. 39.

As a consequence of this decades-long history, free reusability of APIs in furtherance of unhindered and widespread interoperability has become the *sine qua non* of the idea-expression dichotomy in the software context, allowing software to build upon software, platform upon platform, feature upon functionality, in a dynamic competitive environment producing unparalleled innovation.

Section 102(b), and its concomitant interoperability imperative, is one of the only meaningful dividing lines between patents and copyrights with respect to software. The ideas—the methods of operating software functions—should be subject to private ownership and control only if the rules of patent grant such protection. Entitlement to a patent requires proof of eligibility before a government gatekeeper, to assess whether the matter to be patented is sufficiently novel and inventive. No such gatekeeper exists in the copyright realm,

as copyright protection is automatic. Patent protection lasts 20 years; copyright protection can endure for more than a hundred years, and is trivial to secure.

Lotus v. Borland: a method of operating software cannot be protectable, even if expressive and creative

The Java APIs are the rules of a game, the instructions that developers must follow to create and run Java-compatible programs. They articulate how to organize functions in the hierarchical structure of packages, methods and classes. They are the equivalent of the accounting system at issue in *Baker v. Selden*. These labels, and the way Sun classified them within the package/class/method directory structure, are "a process, system, or method of operation" of organizing and labeling software programming functions.

Impressed by the length, originality, and creativity of the Java APIs, the Federal Circuit nevertheless ruled, as a matter of law, that methods of operation that are expressive, creative, or original are entitled to protection. "[W]e conclude that a set of commands … may contain expression that is eligible for copyright protection," the Federal Circuit proclaimed, "as long as the author had multiple ways to express the underlying idea."[20]

[20] *Oracle II*, 750 F.3d at 1367.

In so ruling, the CAFC had to confront directly conflicting precedent, the 1995 *Lotus v. Borland* decision, in which the First Circuit Court of Appeals held that even expressive and creative software interfaces are nevertheless uncopyrightable methods of operation.[21] In that case, Borland had copied the commands the Lotus spreadsheet used so that end user-developed macros (i.e., mini-programs) written for the Lotus spreadsheet program could be readily deployed for use with the Borland spreadsheet program,[22] much like Google emulated a subset of the Java APIs to make it easier for Java applications to be ported to Android. The First Circuit denied copyright protection for these commands, ruling that even "expressive" command structures and labels used in a software application that must be adopted in order to secure interoperability with that software nevertheless constitute a method of operation.[23]

> The fact that there may be … many different ways to operate a computer program using a set of hierarchically arranged command terms, does not make the actual method of operation chosen copyrightable; it still functions as a method for operating the computer and as such is uncopyrightable.[24]

[21] Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807 (1st Cir. 1995), *affirmed without opinion by an equally divided Supreme Court*, 516 U.S. 233 (1996).

[22] *Lotus*, 49 F.3d at 810.

[23] *Id.*

[24] *Id.* at 818.

The "expressive choices" Lotus made in writing those commands were deemed irrelevant. "The fact that Lotus developers could have designed the Lotus menu command hierarchy differently", the *Lotus* court held, was "immaterial" under Section 102(b).[25] Because there is no meaningful difference between the "hierarchically arranged command terms" in the Lotus spreadsheet program and those contained in the Java platform, the Federal Circuit was compelled to explain how *Lotus* is not applicable.

[25] *Lotus*, 49 F.3d at 816.

The CAFC attempted to distinguish the *Lotus* case on three grounds. "First, while the defendant in *Lotus* did not copy any of the underlying code," the court stated, "Google concedes that it copied portions of Oracle's declaring source code verbatim."[26] Google and Borland copied the same thing. Borland copied the labels and command structures of the Lotus spreadsheet program. Google copied the labels and command structures of the Java platform. Just as in the *Oracle* case, "Borland [read, Google] did not copy any of Lotus's [i.e. Sun's] underlying computer code; it copied only the words and structure of Lotus's [i.e. Oracle's] … command hierarchy."[27]

[26] *Oracle II*, 750 F.3d at 1365.

[27] *Lotus*, 49 F.3d at 810.

"Second," the CAFC continued, "the *Lotus* court found that the commands at issue there … were not creative, but it is undisputed here that the declaring code and the structure and organization of the API packages are both creative and original."[28] This statement is false. The *Lotus* court never ruled on whether the commands at issue were creative; their creativity was irrelevant. Because the Lotus command structure was an essential method of operating the Lotus spreadsheet program, the *Lotus* court did "not inquire further whether that method of operation could have been designed differently. The 'expressive' choices of what to name the command terms and how to arrange them do not magically change the uncopyrightable menu command hierarchy into copyrightable subject matter."[29]

[28] *Oracle II*, 750 F.3d at 1365.

[29] *Lotus*, 49 F.3d at 816.

Which brings us to the CAFC's third and final basis for distinguishing *Lotus*: "while the court in *Lotus* found the commands at issue were 'essential to operating' the system, it is undisputed that—other than perhaps as to the three core packages—Google did not need to copy the structure, sequence, and organization of the Java API packages to write programs in the Java language."[30]

Judge Alsup's factual finding, not challenged or even questioned on appeal, was that "the method specification as set forth in the declaration *must be identical* under the Java rules (save only for the choices of argument names). Any other declaration would carry out some *other* function. The declaration requires precision."[31] Google was in fact required to copy the Java APIs in the manner it did in order to secure the level of interoperability Google sought. And as demonstrated below, the level of interoperability Google achieved was significant.

[30] *Oracle II*, 750 F.3d at 1365. The three core packages the CAFC mentions here are significant. *See infra* n. 37-41 and accompanying text.

[31] *Oracle I*, 872 F. Supp. 2d at 998. Because "the rules of Java dictate the precise form of certain necessary lines of code called declarations," Judge Alsup found, "Android and Java must be identical when it comes to those particular lines of code." *Id.* at 979.

Any method of operation can be creative and original; anyone developing methods of operating software functions will have an infinite variety of creative choices to make. By making creativity and originality the only effective touchstone for assessing copyrightability, the CAFC has essentially extended copyright protection "to the methodology or processes adopted by the programmer," contrary to congressional intent in enacting Section 102(b).[32]

[32] "If a framework of shortcuts used by programmers in their development process isn't a procedure, system, or method of operation, what is?" Jonathan Band, *Further Reflections on Oracle v. Google*, Project Disco, May 12, 2014.

The only way to convey the Java rules is to express them verbatim, merging expression with ideas

Even if a method of operating software is sufficiently original or creative so as to justify extending protection to it, the longstanding case law in this area, known as the "merger" doctrine, instructs that any expressive content that exists in the Java APIs has been effectively merged with the facts and ideas those APIs communicate. In the CAFC's own words:

> Under the merger doctrine, a court will not protect a copyrighted work from infringement if the idea contained therein can be expressed in only one way. For computer programs, "this means that when specific [parts of the code], even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement."[33]

[33] *Oracle II*, 750 F.3d at 1360 (citations and quotations omitted).

The Federal Circuit rejected the applicability of the merger doctrine (with one notable exception as described below), because, in its view, merger is relevant only for determining if expression is merged with the idea at the time of the original work's creation, and not for determining if a later work infringed that expression. "Merger cannot bar copyright protection for any lines of declaring source code unless Sun/Oracle had only one way, or a limited number of ways,

to write them", and the factual record, the CAFC observed, demonstrated that Sun had "unlimited options as to the selection and arrangement" of the Java APIs.[34] For example, with respect to the "math.max" API highlighted above, the Federal Circuit stated that Sun's developers could have called that API "any number of things, including 'Math.maximum' or 'Arith.larger.'"[35]

[34] *Oracle II*, 750 F.3d at 1361.

[35] *Id.*

That Sun had unlimited options in its creation of the Java platform is self-evident. But for anyone else seeking to develop software that is interoperable with that platform, the options are quite limited. The functionality for comparing two integers must be called "math.max" (and must be included in the package java.lang) if the Android platform is to be compatible with the Java platform. Because "the rules of Java dictate the precise form of certain necessary lines of code called declarations," Judge Alsup found, "Android and Java must be identical when it comes to those particular lines of code."[36]

[36] *Oracle I*, 872 F. Supp. 2d at 979.

Curiously, the Federal Circuit incongruously recognized this in the portion of its opinion discussing three "core" Java package APIs. These three packages, Judge Alsup explained, were deemed core to the Java programming language from its inception:

> When Java was first introduced in 1996, the API included eight packages of pre-written programs. At least three of these packages were "core" packages, according to Sun, fundamental to being able to use the Java language at all. These packages were java.lang, java.io, and java.util. *As a practical matter, anyone free to use the language itself (as Oracle concedes all are), must also use the three core packages in order to make any worthwhile use of the language.*[37]

[37] *Oracle I*, 872 F. Supp. 2d at 982 (emphasis added).

With respect to these packages, the CAFC stated:

> It seems possible that the merger doctrine, when properly analyzed, would exclude the three packages identified by the district court as core packages from the scope of actionable infringing conduct. *This would be so if the Java authors, at the time these packages were created, had only a limited number of ways to express the methods and classes therein if they wanted to write in the Java language.* In that instance, the idea may well be merged with the expression in these three packages.[38]

[38] *Oracle II*, 750 F.3d at 1362 (emphasis added). Unhappily for Google, the CAFC nevertheless ruled against Google on its merger argument as to these three core packages because "Google did not present its merger argument in this way below and … does not try to differentiate among the packages for purposes of its copyrightability analysis …." Id.

With this single passage, the CAFC unwittingly revealed the nature of the Java APIs as a method of operating software. Applications that are written in the Java programming language run only on platforms that contain the same Java APIs that the language mandates.[39] At least with respect to these three core packages, the CAFC conceded that the dictates of the Java programming language constrained Sun's choices in developing a platform that faithfully executes applications written in the Java language. If so, these APIs are not entitled to protection; the ideas or facts posed by the Java programming language have merged with the expression of those APIs.

[39] Java is both an execution environment on which Java applications run, and a programming language for writing such applications. Oracle Corporation, *What is Java technology and why do I need it?*. Sun's objective was to create a universal application execution environment such that applications written in the Java language can run with no or minimal adaptation across multiple devices and operating systems (such as Windows, MacOS and Linux) and Internet browsers (such as Chrome, Explorer and Safari), so long as such devices, systems and browsers contain a ported version of the Java execution environment. In short, Sun sought application "write once, run anywhere" portability for Java applications. *Oracle I*, 872 F. Supp. 2d. at 977. Ironically, Java itself was enabled by the open or non-proprietary nature of the APIs of the operating systems and browsers to which the Java platform has been ported. To make Java successful, Sun required widespread adaptation of the Java platform to multiple computer and device operating systems and Internet browsers, which in turn required access to the APIs of those systems and browsers.

Yet under the CAFC's ruling, Oracle is somehow entitled to prevent Google's use of the other Java APIs at issue, all of which are no less constrained by the dictates of the Java programming language.[40] The CAFC evidently failed to grasp that (1) Android applications are written in the Java programming language;[41] and (2) all Java APIs, including those at issue in the case, are derived from and require adherence to the rules of the Java programming language. This critical connection between the Java programming language and the Java platform was apparently lost on the Federal Circuit.

[40] *See What are the 37 Java API packages possibly encumbered by the May 2014 Oracle v Google decision?*, StackOverflow (Question: "How can I avoid using the encumbered APIs [found by the CAFC to be infringed by Google] in my Java code?" Answer: "*Don't write in Java. Anything written in the Java programming language will involve classes from the affected packages*." (emphasis added)).

[41] *See infra* n. 47-51 and accompanying text. Throughout the CAFC's opinion, the court made statements that seemingly betray an ignorance of the fact that Java (and Android) applications are written in the Java programming language *so that such applications can run on a Java-compatible platform*. In nearly every instance in which the court mentions the language, it does so in the context of Google's or Sun's design of the Android or Java platforms and APIs. *See e.g., Oracle II*, 750 F.3d at 1350 (referencing Google "using the Java programming language" to design the Dalvik virtual machine (i.e., Google's version of the Java platform, which is in fact written in the C language)); *id.* at 1362 ("This would be so if the [Sun/Oracle] Java authors [of the Java APIs], at the time these packages were created, had only a limited number of ways to express the methods and classes therein if they [the authors of the APIs] wanted to write in the Java language."); *id.* at 1363 ("Google could have written its own API packages using the Java language. Google chose not to do that."); *id.* at 1365 ("Google did not need to copy the structure, sequence, and organization of the Java API packages *to write programs in the Java language*." (emphasis added)). No wonder the CAFC found Google's interoperability arguments "confusing." *Id.* at 1371.

Interoperability is the foundational rationale behind Section 102(b) in the software context

The most troublesome aspect of the CAFC's ruling is its pronouncement that interoperability has nothing to do with the copyrightability question. "Google," the court noted with less-than-circumspect suspicion, "chose to copy Oracle's declaring code and the [structure, sequence and organization of it] to capitalize on the preexisting community of programmers who were accustomed to using the Java API packages. That desire has nothing to do with copyrightability."[42]

[42] *Oracle II*, 750 F.3d at 1372.

If the task at hand is to determine if something in software is an uncopyrightable method of operation, examining how and why a party other than the originator of that method implemented it is an integral part of the Section 102(b) analysis. "Seeking to capitalize on a pre-existing community of programmers" is of course seeking to promote interoperability, and interoperability "sheds further light on the character of the command structure as a system or method of operation."[43] Because promoting software interoperability has become the policy foundation underlying Section 102(b), we need to know if Google was simply trying to play the same game that Sun was playing, or was taking more Sun-originated material than was necessary to play.

[43] *Oracle I*, 872 F. Supp. 2d at 1000. *See also* Jonathan Band, *Further Reflections on Oracle v. Google*, Project Disco, May 12, 2014 ("The Federal Circuit opined that the desire to capitalize on the preexisting community of Java programmers 'has nothing to do with copyrightability.' But this plainly is wrong. What could be better proof that something is a procedure, system, or method of operation than if a person can become 'trained,' 'experienced,' or 'accustomed' to using it in the course of developing new works?").

The reason the CAFC was so willing to dismiss Google's interoperability-based arguments is because the court failed to discern any interoperability benefit at all as a result of Google's actions.[44] The Federal Circuit repeatedly stressed in its opinion that "Android is not generally Java compatible",[45] and that "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result."[46]

[44] Another reason Google's interoperability objectives, and its success, were so heavily discounted by the Federal Circuit is because the court erroneously consigned interoperability concerns to the hopelessly fact-specific and unpredictable fair use swamp. *Oracle*, 750 F.3d at 1372. Patently hostile to Google's fair use contentions, the CAFC came alarmingly close to ruling in Oracle's favor as a matter of law. *Id.* at 1376. A robust critique of the CAFC's fair use explication, and its irreconcilability with the Ninth Circuit's *Sony* and *Sega* decisions, is certainly possible and imperative, but outside the scope of this article. *See infra* n. 52.

[45] *Oracle II*, 750 F.3d at 1351.

[46] *Id.* at 1361.

The CAFC's ruling on this point flatly contradicts the factual record of the case and is, more importantly, entirely divorced from the reality of Android application development. As Judge Alsup concluded:

> *Although the declarations must be the same to achieve the same functionality*, the names of the methods and the way in which the methods are grouped do not have to be the same. Put differently, many different API organizations could supply the same overall range of functionality. *They would not, however, be interoperable.*[47]

[47] *Oracle I*, 872 F. Supp. 2d at 982 (emphasis added).

Despite that the Federal Circuit found "no evidence in the record" of any Java applications "that either pre-dated or post-dated Android that could run on the Android platform",[48] every Java application ever developed now or in the future can more easily be migrated to Android because of Google's adoption of the Java APIs in question. The result of Google's actions is that a large number and variety of programming tools exist to support Android application development in Java. These tools leverage the commonality in the two platforms to make it easy for Java programmers to write Android applications.[49]

[48] *Oracle II*, 750 F.3d at 1371.

[49] Alex Marshall, *Top 10 Android Apps and IDEs for Java Coders and Programmers*, Java PDF Blog, Dec. 11, 2014.

And unmistakably, Google's interoperability efforts in this respect have been a resounding success. Java is the go-to programming language for developing Android applications.[50] Much to Oracle's chagrin, the emergence of Android has in fact led to a resurgence in the popularity of the Java programming language:

> After a year-and-a-half in second place, behind the C language, Java surged back into first place in this month's Tiobe language popularity index. *Topping the Tiobe Index again is being attributed to Java's usage in Android application development*.[51]

[50] *See e.g.*, Laurence Bradford, *Five Things to Know before Building Your First Android App*, Learn to Code with Me, Aug. 25, 2014 ("You need to learn Java, there's no way around it."); Shane Conder & Lauren Darcey, *Learn Java for Android Development*, TUTS+, Sept. 13, 2010 ("Android applications are developed using the Java language. As of now, that's really your only option ….").

[51] Paul Krill, *Java Regains Spot as Most Popular Language in Developer Index,* InfoWorld, Apr. 14, 2015.

Google's actions and Oracle's response encapsulate the character of the Java APIs as a method of operation. Google impinged on Oracle's attempted monopolization of the instructions for writing and supporting Java programs on mobile devices, and so Oracle sued to stop Google from capitalizing on the existing Java developer base. In essence, Oracle sought exclusivity over that developer community. It's the same impulse that motivated Sony and Sega to incur significant legal expense in an attempt—ultimately unsuccessful—to shut down the efforts of companies that made it possible to play Sony Playstation games on a personal computer, and Accolade PC games on Sega's proprietary

game console.[52] In the apt words of Judge Alsup, "fragmentation, imperfect interoperability, and Oracle's angst over it illustrate the character of the command structure as a … method of operation."[53]

[52] Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); Sony Computer Entertainment, Inc. v. Connectix Corp., 203 F.3d 596 (9th Cir. 2000). The facts of the *Sega* case should be familiar to followers of the Oracle v. Google proceedings. In the early nineties, Sega manufactured video game consoles and game cartridges that could be used only with the Sega console. Sega licensed the code necessary to produce Sega-compatible game cartridges to game publishers. Accolade, an independent game publisher, attempted to negotiate a license with Sega, but could not accept Sega's demand that Sega be the exclusive manufacturer of all games produced by Accolade. *Sega*, 977 F.2d at 1514-16. Accolade therefore reverse engineered the Sega console and game cartridges in order to discover the "interface specifications"—that is, the APIs or, in the terminology of the Ninth Circuit's opinion, "header files"—that enabled Accolade's Mac and PC games to run on—that is, interoperate with—the Sega console platform. *Id.* Intermediate copying as an integral part of reverse engineering, the *Sega* court held, is not actionable if the point of that effort is to discover the interfaces needed for compatibility. *Id.* at 1527. Concededly, the *Sega* court's ruling was based exclusively on fair use—but its decision was undeniably premised on the unprotectable nature of interfaces as methods of operation under Section 102(b). *Id.* The more recent *Sony* decision is in accord. *Sony*, 203 F.3d at 602 ("Connectix's intermediate copying and use of Sony's copyrighted BIOS was a fair use for the purpose of gaining access to the unprotected elements of Sony's software."). Notably, the fact that Connectix failed to achieve perfect compatibility was not material. *Id.* at 599.

[53] *Oracle I*, 872 F. Supp. 2d at 1000.

Because Google failed to achieve perfect compatibility, an impossibility given the differences between a platform developed for desktop computers (Java) and a platform for mobile devices (Android), the CAFC was convinced that Google's actions could not be explained by a desire to achieve interoperability. As a result, the CAFC discounted every argument Google raised regarding Section 102(b), interoperability, and even fair use. The unfortunate logic of the Federal Circuit's analysis invites courts to assess degrees of compatibility in future cases.[54]

[54] As Judge Alsup noted with some irony, "Oracle has made much of [the fragmentation problem], at times almost leaving the impression that if only Google had replicated *all* 166 Java API packages, Oracle would not have sued." *Oracle I*, 872 F. Supp. 2d at 1000 (emphasis in original).

The strawman that APIs are not copyrightable because they are "functional"

One of the more frustrating aspects of the Federal Circuit's decision is its mischaracterization of Judge Alsup's holding:

> [T]he district court recognized that the SSO [(the structure, sequence and organization of the Java APIs)] "resembles a taxonomy," but found that "it is nevertheless a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned functions." *In other words, the court concluded that, although the SSO is expressive, it is not copyrightable because it is also functional.* The problem with the district court's approach is that computer programs are by definition functional—they are all designed to accomplish some task.[55]

[55] *Oracle II*, 750 F.3d at 1367 (emphasis added).

Judge Alsup's holding was not that the Java APIs are unprotectable because they are functional. They are in fact not functional.[56] They are declarative; they dictate how functional elements should be organized and labeled. Rather, Judge Alsup's holding was that the Java APIs are unprotectable because they are a method of operating Java software programs, a method that can only be implemented in the same manner as implemented in the replicated platform in order to achieve the entire point of the exercise—interoperability.

[56] "The court seems to not understand what an API is, confusing it with software *functionality*." Mike Masnick, Tech Dirt, May 9, 2014, *Appeals Court Doesn't Understand the Difference between Software and an API; Declares APIs Copyrightable* (emphasis in original). Admittedly, the *Sega* and *Sony* courts used the "functional" label as a shorthand means of referring to methods of operation barred by Section 102(b) (Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510, 1517 (9th Cir. 1992); Sony Computer Entertainment, Inc. v. Connectix Corp., 203 F.3d 596, 603 (9th Cir. 2000)), but neither court's holding, nor Judge Alsup's, hinged on the functional or non-functional (i.e., behavioral) nature of the interfaces in question.

Having mischaracterized Judge Alsup's holding, the appellate court then gratuitously ran with it, accusing Judge Alsup (who is a coder himself)[57] of applying a rule that would result in all computer programs losing copyrightability. In the words of the CAFC, "[i]f we were to accept the district court's suggestion that a computer program is uncopyrightable simply because it 'carr[ies] out pre-assigned functions,'" a holding Judge Alsup never made, "no computer program is protectable."[58] This accusation is ironic, given that the CAFC's reasoning inexorably leads to all methods of operation within software being protectable.

[57] Mike Masnick, *Should People Learn to Code? Yes if they are Judges Ruling on Cases Involving Software*, Techdirt, May 21, 2012. That Judge Alsup understood the tension between extending copyright protection to software and yet denying protection to "methods of operation" is demonstrated by footnote 7 of the court's opinion, in which Judge Alsup notes that a prominent software copyright academic "has argued that the Section 102(b) terms 'process,' 'system,' and 'method of operation' should not be understood literally for computer programs." *Oracle I*, 872 F. Supp. 2d at 996 n.7, *citing* Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of Computer Software*, 94 Colum. L. Rev. 2559, 2569-70 (1994) ("[S]ome ways of achieving 'a certain result' (i.e., some 'behaviors') will be protected by copyright. What ways, and what results?")

[58] *Oracle II*, 750 F.3d at 1367. The CAFC took another cringeworthy whack at this straw man, and doubled down on its factual error regarding the choices available to Google given Google's legally-protected interoperability objectives, in a concluding paragraph notable for its succinct distillation of the myriad legal and factual flaws of this decision. "Given the court's findings that the [structure, sequence and organization of the Java APIs] is original and creative, and that the declaring code could have been written and organized in any number of ways and still have achieved the same functions, we conclude that Section 102(b) does not bar the packages from copyright protection just because they also perform functions." *Id.* at 1368.

Again, we are concerned here with nothing more than the magic words that Sun used for labeling, classifying and invoking functionality. We are focused on the declarations that take the form: java.package.class.method(). No functionality, let alone protectable expression of functionality, was copied; Google developed its own.

Judge Alsup's concluding passage in his copyrightability decision captures the point perfectly:

> In closing, it is important to step back and take in the breadth of Oracle's claim. Of the 166 Java packages, 129 were not violated in any way. Of the 37 accused, 97 percent of the Android lines were new from Google and the remaining three percent were freely replicable [declarations] …. Oracle must resort, therefore, to claiming that it owns, by copyright, the exclusive right to any and all possible implementations of the taxonomy-like command structure for the 166 packages and/or any subpart thereof—even though it copyrighted only one implementation. *To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands. No holding has ever endorsed such a sweeping proposition.*[59]

[59] *Oracle I*, 872 F. Supp. 2d at 1001-02 (emphasis added).

Unfortunately, the Federal Circuit has done just that, and now we know that the Supreme Court is not going to correct it. Oracle has effectively secured, for decades, a patent-like monopoly over a method of operating software.

The Java APIs may have been time-consuming and expensive to develop, and there may have been some creative choices involved. Nevertheless, they are "the rules by which computer programs interact with one another, analogous to the rules of games played by humans, like the rules of checkers or chess, which are agreed by all to be uncopyrightable ideas."[60] Under the CAFC's logic, no method of operation will ever be excluded so long as the author of that method exercised creativity. If the only criteria for extending copyright protection to software is originality and creativity, Section 102(b) loses all meaning.

[60] Brief for Software Freedom Law Center and Free Software Foundation as *Amici Curiae* Supporting Respondent at 8-9, Google Inc. v. Oracle America, Inc. No. 14-410 (U.S. Sup. Ct. Dec. 8, 2014).

"Google could have written its own API packages using the Java language",[61] the CAFC declared, but failed to explain what the point of that would be, if not to enable compatibility between Java and Android. In developing Android to be interoperable with the Java platform, Google was playing the Java game using some, but not all, of the rules that Sun established. Yet, Google made Android sufficiently familiar to application developers accustomed to writing applications in Java as to unleash thousands of applications on Android in short order. This is an outcome that copyright law should never prevent.

[61] *Oracle II*, 750 F.3d at 1353.

Copyright protection for software, even for content that is expressive, original or creative, must be denied if that same content constitutes nothing more than a description and classification of functionality. Just as the originator of the accounting system at issue in *Baker v. Selden* was not able to use copyright law

to prevent anyone else from adopting the accounting methods and forms he described in his book, Oracle must not be allowed to use copyright to preclude anyone else from following the same methods and systems described in the Java APIs. If the idea-expression dichotomy in the software context is to have any significant impact in limiting the scope of copyright protection, it must be that copyright extends only to code that is completely original, and that is not dictated by the requirements of compatibility with an external standard.

The software industry, and indeed every industry that relies on software, has thrived for decades without the encumbrances of proprietary claims over APIs. Because the Federal Circuit's decision destroys the balance between copyrightable expression and uncopyrightable ideas in software, it threatens competition and innovation. The Ninth Circuit should repudiate it at the earliest opportunity.