Intellectual Property & Technology Law Journal

Edited by the Technology and Proprietary Rights Group of Weil, Gotshal & Manges LLP

VOLUME 33 • NUMBER 5 • MAY 2021

Software Copyrightability: Where We Are and What's to Come

Christopher M. Bruno and Darra S. Loganzo

> **"[T]**he essential, threshold proposition stands: by bringing software into the world of copyright, Congress plainly did not mean to *abrogate* longstanding copyright principles; it meant to subject software to them." – Ralph Oman, Former U.S. Register of Copyrights[1]

Computer software holds a unique seat among works of art that receive copyright protection under the federal Copyright Act. The digital commands and processing instructions that comprise software seemingly do not fit the mold of traditional "artwork" as sculptures or songs do. But a review of court approaches to software copyrightability reveals that courts have steadfastly attempted to use the same lens and methodology of analyzing software as with other forms of art. Courts subject software to tests that, at their core, ask the same basic copyright questions: is the element functional or expressive, original or commonplace?

Despite the application of the Copyright Act to software for decades, there is surprisingly significant uncertainty about what elements are actually protectable. In the seminal copyright software case from the U.S. Court of Appeals for the Second Circuit, *Computer Associates International, Inc. v. Altai, Inc.*, the court admitted that "[t]o be frank, the exact contours of copyright protection for non-literal program structure are not completely clear" but expected "that as future cases are decided, those limits will become better defined."[2] They are not.

Instead, for the past three decades, district and circuit courts have avoided a clear-cut boundary line for the copyrightability of the elements of computer software, and reluctant to clarify those limits, adhere to the same dichotomy applied to other works.

As the *Altai* court's expectation remains unfulfilled, the technology industry expands, and new types of software become integrated into daily life, the question continues to build: Where is the line for software copyrightability and how will courts continue to make their decisions?

GENERAL ANALYSIS OF COPYRIGHTABILITY

Courts typically see questions of copyrightability arise in infringement actions, where a copyright

Christopher M. Bruno is a partner at McDermott Will & Emery managing a broad range of intellectual property litigation cases, including patent-related matters in pharmaceuticals, computing, and biomedical industries. He also handles copyright matters and related contract disputes. **Darra S. Loganzo** is an associate in the firm's intellectual property litigation group, focusing on patent, trademark, and copyright matters. The authors may be reached at *cbruno@mwe.com* and *dloganzo@mwe.com*, respectively.

holder accuses a defendant of copying a work without permission. The settled elements of a copyright infringement action ask the trier of fact to examine whether the plaintiff owns a valid copyright on the work, the defendant copied the plaintiff's work, and the defendant's work is "substantially similar" to the plaintiff's protected work.[3]

Focusing on the third prong, courts have applied various tests to describe "substantial similarity," depending on the jurisdiction and type of work at issue. When a work contains both protectable and unprotectable elements, courts must identify whether the protectable elements are substantially similar – an inquiry into the components' copyrightability.[4]

Particularly in software, the tasks of understanding which portions are protectable and defining each portion's purpose – i.e., separating the functional (unprotected) from the expressive (protected) – are both necessary and difficult. Under copyright law, functional works (or parts of works) are not copyrightable because they are not original expression. For example, if there is only one way for software to complete a task, the implementation of that task is functional, and the task cannot be protected by copyright.[5]

By contrast, when there are alternative and creative choices that an author can make to develop a product and reach the same result, those decisions are considered creative expression and thus are protected by copyright.[6]

Currently, in most federal circuit courts of appeals, courts analyze substantial similarity of software through the lens of the three-step abstraction-filtration-comparison ("AFC") test, first set forth in *Computer Associates International, Inc. v. Altai, Inc.*[7]

The Second, Fifth, Eighth, Ninth, Tenth, and Eleventh Circuit Courts of Appeals have each adopted the AFC test,[8] and the Sixth Circuit applies a similar test.[9]

The First and Seventh Circuits have favorably viewed the *Altai* test as useful for assessing copying of non-literal elements, but have criticized the AFC test as "misleading" for literal elements.[10]

The Fourth Circuit has not explicitly endorsed the test, but at least one district court in the Fourth Circuit has employed it.[11]

By contrast, the Third Circuit stands alone in applying the *Whelan* test – a test that *Altai* sought to displace and other circuits have expressly rejected. The *Whelan* test focuses on the "end sought to be achieved by the work" and whether there is any other way to achieve that purpose or function.[12] Thus, the AFC test remains dominant throughout the United States.

Courts employing the AFC test dissect a software program into its structural elements. The three steps are abstract the components that are protected and unprotected; filter out the unprotected components; and compare the protected components of the copyrighted and infringing work to see if infringement occurred.[13]

Under the abstraction step, courts break down a computer program's constituent components into a hierarchical structure to separate the unprotectable utilitarian functions from the protectable expressive components.[14] Then, under the filtration step, courts separate the protectable expression from the unprotectable material at each level of abstraction. This "unprotectable" material includes ideas, elements from the public domain, or elements dictated by efficiency considerations or external factors.[15]

Last, under the comparison step, courts compare the remaining protectable elements to the allegedly copied work to determine if the protectable parts of the works are substantially similar.[16] By applying this methodology, the substantial similarity inquiry focuses on comparing only similarities between the remaining "gold nugget[s]" of the work's copyright value.[17]

In addition, courts often consider two other doctrines as part of the software copyrightability analysis because they both embody the idea-expression dichotomy: merger and *scenes a faire*.

Under the merger doctrine, when unprotected and protected components of an artwork are inseparable and essential, they are inextricably intertwined and merge into an unprotectable work as a whole.[18] This doctrine ensures that only creative expression is protected, rather than the underlying idea, process, or system.[19] The merger doctrine is frequently used as a defense to copyright infringement, where a defendant may argue that the work is not copyrightable because the idea can be expressed in only a limited number of ways. Without the merger doctrine, copyright protection

could be used to inhibit innovation of functional software improvements.

Next, under *scenes a faire*, a concept meaning "scenes that must be done," themes that are merely standard or customary in a genre are not protected by copyright. In the software context, this means that industry standards and practices that are commonplace to solve routine problems or "practical realities" are uncopyrightable.[20] Applying *scenes a faire*, courts have found that elements of a software program that are "dictated by external factors" such as hardware standards, mechanical specifications, consumer demands, and compatibility requirements are ineligible for copyright protection.[21] This doctrine often arises because, as a function of the medium, software development frequently reuses certain basic computing commands. Thus, the prevalent use – or overuse – of a command can transform it into an industry standard, barring it from copyright protection because of its ubiquity.

Nothing about these tests is new; rather, they are well settled in law that had developed many decades ago related to other instruments that have both functional and creative components. Back in 1954, the Supreme Court held in *Mazer v. Stein* that the expressive and aesthetic design of a lamp's base can be copyrightable, even when the lamp serves the broader function of lighting a room.[22]

Similarly, instruction manuals and handbooks are protectable to the extent they utilize creative literary choices to describe an underlying unprotectable process.[23] Thus, at least so far, the analysis of the combined functional and expressive elements in software mirrors the traditional process of determining copyrightability. As further demonstrated below, there is still no blueprint for courts to follow when analyzing which elements of computer software will be found to be creative and protectable.

Applying the Tests to Software

Courts have generally divided the creative, and thus potentially copyrightable, elements of software into two categories: literal and non-literal.

Literal elements are the lines of written source code and object code, and non-literal are everything else – i.e., the aspects not reduced to code, such as the overarching themes or experiences, imagery, narrative, and the look and feel of the user interface including menu structures and screen displays.[24] Even when the literal text of a code is not copied, courts can still find copyright infringement when "the fundamental essence or structure" of a work is duplicated.[25]

Courts have applied the AFC test for both literal and non-literal elements to determine substantial similarity. For literal elements, courts look at the "line-by-line" text of the source code and object code.

Alternatively, analyzing non-literal elements requires a more holistic approach where courts consider the overall feel of the software and the decisions that were made to achieve the program's goal that are conveyed in forms beyond just the text of the code. The Ninth Circuit has recognized software to have both "individual non-literal elements," which include discrete visual and audible components, such as a website's logos, fonts, and sound effects, and also "dynamic non-literal elements," encompassing the real life experience and operation of the software, such as a website's interactive presentation of the content.[26]

Applying the AFC test, courts begin with the lowest level of generality, the line-by-line code itself. At the highest levels of abstraction are theoretical ideas and the conceptual functions the code is seeking to achieve. A computer program can often be parsed into at least six levels of abstraction (beginning from the lowest literal elements, to the higher non-literal elements): object code; source code; algorithms and data structures; modules; the program structure or architecture; and the program's main purpose.[27]

Courts have found non-literal elements of a software's structure, sequence, organization, ("SSO") and user interface to be protectable.[28] For example, the Fifth Circuit has explicitly noted that "abstraction of non-literal elements of computer programs reveals a 'spectrum' of copyrightable material, ranging from uncopyrightable simple forms to 'high expression,'" with a website's menu structures as an example that falls within those two extremes.[29]

Courts have also instructed that non-literal elements of a software's screen display can be protectable original expression, if the display is distinguishable from purely functional menu commands.[30]

Recent Trends in Software Copyrightability

While courts have not developed a bright-line rule that separates the spectrum into protectable and unprotectable elements, recent case law

reflects district courts coalescing on some standards. Functional, unprotected components of software include: open source public elements, facts and data, mathematical and statistical functions, processes, systems, and methods, well-known conventional displays, short phrases, constants and coefficients, formulas and algorithms, architecture and modules, menu commands, test results, and data structures.[31]

The commonality of the unprotected elements is their copy-and-paste nature; they are structures that are merely reused and repeated across the industry. Just as an alphabet letter cannot be copyrighted until it is arranged to form an original literary piece, so too the internal backbones that comprise computer software cannot be copyrighted until arranged, combined, and varied into an original and expressive work.

Yet, not all hope is lost in protecting software incorporating these otherwise functional elements. Rather, they can be selected and arranged, which transforms them into something more – a copyrightable work.[32] *M-I L.L.C. v. Q'Max Solutions, Inc.* is illustrative.

There, the plaintiff sought to distinguish among Graphic User Interface ("GUI") aspects of the interface screen.[33] The court considered whether the menu naming, organization, and labeling are merely industry standards for developers, or if there was a creative selection, arrangement, and variation of presentation, naming, and coloring.

The court ultimately found that using an "expandable tree" to display a menu is "not an original choice" after considering evidence that this organizational structure is a "basic" style.[34] By contrast, the arrangement and presentation of the header bar and graphs had some variation in presentation and coloring that was "left to the discretion of the program's author."[35] The court's analysis reveals that whether a choice is creative depends on how much the author diverges from industry need and norm.

With a similar perspective in mind, in *Datacarrier S.A. v. Woccu Services. Group*, the court found that switch-to-host message format software is "routinely functional" because it uses only simple number fields derived from industry standards for financial transactions and the design was based on "functional and efficiency concerns."[36]

Notably, the fields were primarily selected for functional reasons, not expressive value. The field characteristics, such as their length for inputs, were determined based on efficiency of processing the request: to accommodate a certain number of characters but not be wasteful.

Moreover, the order of selection of the fields were found to be merely trivial and arbitrary, and the fields viewed as "short, simple, fillable" blank forms to be filed for processing the transaction, rather than for any creative rationale.

Despite these recent trends, uncertainty remains. The state of the law remains murky and fact-specific, eschewing the certainty that clearly delineated rules for different software elements would produce. Under the current framework, it is difficult for software developers or programmers to know in advance which elements of their work are copyrightable. Instead of waiting and guessing, better guidance is necessary as software continues to increase in complexity.

RECENT SUPREME COURT OPINION: *GOOGLE V. ORACLE*

Many hoped that the U.S. Supreme Court's decision in *Google LLC v. Oracle America, Inc.* might provide much needed instruction on software copyrightability. Unfortunately, no such guidance followed in the Court's decision issued on April 5, 2021.[37]

The Court had granted certiorari to determine two issues: (1) whether copyright protection extends to Oracle's Java application programming interface ("API"), and if so, (2) whether Google's use of the API was permitted under the doctrine of fair use.[38] As background, API is a type of "declaring code" that is a library of functions and commands composed of prewritten packages, classes, and fields to allow for a common platform for computers to communicate with each other.

Rather than address the first issue – whether the software is ineligible for copyright protection when there is only one way to complete the function of the code – the Supreme Court assumed that API is copyrightable:

> A holding for Google on either question presented would dispense with Oracle's copyright claims. Given the rapidly changing technological, economic, and business-related circumstances, we believe we should not answer more than is necessary to resolve the parties'

dispute. *We shall assume, but purely for argument's sake, that the entire Sun Java API falls within the definition of that which can be copyrighted*. We shall ask instead whether Google's use of part of that API was a "fair use."[39]

The remainder of the majority opinion focuses solely on the fair use analysis, and ultimately, the Supreme Court held that Google's use of the code was fair use. Yet, in sidestepping the gateway issue of copyrightability, the Supreme Court missed an opportunity to clarify the threshold question about whether API is copyrightable.

The issue of whether API satisfied software copyrightability requirements was primed for resolution. From Google's perspective, API is just another example of an uncopyrightable functional element, as it is a set of instructions that can be written in only one way and all of the pre-programmed functions necessarily rely on the API code. Google argued that the declarations are akin to a key that fits into a lock[40] and it "would happily not reuse the Java SE declarations if we could," but "the language only permits us to use those."[41] Oracle countered that its API is not functional because it made specific and creative code-drafting decisions that merit protection.[42] According to Oracle, Google could write different code to perform the same function if it desired a similar outcome. There were "unlimited options" to write and organize the code at the outset, and Oracle chose one path in how it created its code, which took enormous efforts and resources.[43] Just because the API instructions are now the most widely used in the industry, Oracle's success should not prevent it from receiving its rightful protection. These arguments represent how the courts' previous holdings in copyright software cases have left gaps for advanced technology that *Altai* did not predict would become a prevailing infrastructure.

While APIs have been around since the early 2000s, their use has proliferated in the past five to 10 years. Indeed, APIs have been widely adopted but do not perfectly fit within the traditional framework of the current copyrightability analysis, as they are neither wholly utilitarian nor merely expressive. Should an API be viewed more like an individual functional element of menu commands, fields, formulas, and algorithms that have been found to be unprotectable? Or if a company like Oracle undertakes sufficient time-consuming decisions and creative steps like those explored in *Feist*, *M-I L.L.C.*, and *Datacarrier* to choose, compile, select, and arrange the unprotectable functional elements then it becomes something expressive and protected?

The copyrightability inquiry presented to the Court could have decided whether API's proper place would be joining the growing list of basic structural elements that are unprotected for their reusable and functional nature, or, whether it is an example of an element with time-consuming and expensive development to render it into a creative form. Instead, this question is left unanswered.

In dissent, Justice Thomas, joined by Justice Alito, criticizes the majority for bypassing this first crucial question.[44] Justice Thomas offers three views on the copyrightability of computer software issue: (1) Although computer code occupies a unique space in intellectual property, straddling the categories of copyright and patent law, Congress intended computer software to be subject to the law of copyright;[45] (2) Congress intended to protect declaring code, including the API at issue, under copyright protection alongside protected implementing code because computer code is defined in the Copyright Act as "statements or instructions to be used directly or indirectly" to bring about a result, and declaring code operates indirectly alongside direct implementing code without a categorical distinction between the two;[46] and (3) Google's argument that declaring code is functional and thus unprotectable fails because there were "innumerable ways" for Oracle to write the code.[47]

Though nonbinding, the dissent is essentially the first (and hopefully not the last) word from the Supreme Court in a continually developing and uncertain area of IP law. It certainly sets the stage for future courts to provide their own musings on software copyrightability in future cases.

WHAT'S TO COME?

Given the Supreme Court's decision not to decide whether API is copyrightable, technology companies, computer programmers, and academics are still left without a compass. But as the issues of software copyrightability percolate, current trends remain instructive. Generally speaking, courts have resisted making bright-line and software-specific rules, instead opting to apply the same foundational principles drawn from

cases in other areas of art. The incremental approach that courts employ, narrowly focusing on elements of the specific software at issue rather than settling longstanding debates, is likely to continue.

For now, it is likely that courts will continue to relegate software cases to a corner of copyright law that it is seemingly outgrowing. As the industry expands, new questions will continue to arise that may force courts to reconsider software's place in copyright law.

Notes

1. Ralph Oman, *Computer Software As Copyrightable Subject Matter: Oracle v. Google, Legislative Intent, And The Scope Of Rights In Digital Works*, 31 Harv. J. of L. & Tech. 639, 640 (2018).
2. *Comput. Assocs. Int'l Inc. v. Altai Inc.*, 982 F.2d 693, 712 (2d Cir. 1992).
3. *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 361 (1991); *Cavalier v. Random House, Inc.*, 297 F.3d 815, 822-23 (9th Cir. 2002).
4. *Williams v. Crichton*, 84 F.3d 581, 587-88 (2d Cir. 1996).
5. *RJ Control Consultants, Inc. v. Multiject, LLC*, 981 F.3d 446, 457-58 (6th Cir. 2020).
6. *Capstone Logistics Holdings, Inc. v. Navarrete*, No. 17 Civ. 4819 (GBD) (BCM), 2020 U.S. Dist. LEXIS 110304, *23-24 (S.D.N.Y. June 23, 2020).
7. *Comput. Assocs. Int'l Inc.*, 982 F.2d at 706-10.
8. *Eng'g Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335, 1342-43 (5th Cir. 1994); *I-Systems, Inc. v. Softwares, Inc.*, No. 02-1951 (JRT/FLN), 2004 U.S. Dist. LEXIS 6001, at *30-37 (D. Minn. Mar. 29, 2004); *Sega Enters. v. Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992) ("In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one."); *Country Kids 'N City Slicks, Inc. v. Sheen*, 77 F.3d 1280, 1284-85 (10th Cir. 1996); *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1545 (11th Cir. 1996).
9. *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d. 522, 534 (6th Cir. 2004); *Kohus v. Mariol*, 328 F.3d 848, 854 (6th Cir. 2003).
10. *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 814–15 (1st Cir. 1995); *Empire Med. Review Servs. v. CompuClaim, Inc.*, 326 F. Supp. 3d 685, 702 (E.D. Wis. 2018).
11. *CSS, Inc. v. Herrington*, No. 2:16-cv-01762, 2017 U.S. Dist. LEXIS 120396, at *15-16 (S.D.W.Va. Aug. 1, 2017).
12. *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222, 1235-46 (3d Cir. 1986); *Southco, Inc. v. Kanebridge Corp.*, 390 F.3d 276, 290-92 (3d Cir. 2004).
13. *Comput. Assocs. Int'l Inc.*, 982 F.3d at 706-08.
14. *Id.* at 706; *Eng'g Dynamics, Inc.*, 26 F.3d at 1343.
15. *Eng'g Dynamics, Inc.*, 26 F.3d at 1343.
16. *Comput. Assocs. Int'l Inc.*, 982 F.3d at 709-10.
17. *Id.* at 710.
18. *Baker v. Selden*, 101 U.S. 99, 102-03 (1879); *Carol Barnhart Inc. v. Econ. Cover Corp.*, 773 F.2d 411, 419 (2d Cir. 1985).
19. 17 U.S.C. § 102(b); *CCC Info. Servs. v. MacLean Hunter Mkt. Reports, Inc.*, 44 F.3d 61, 68-70 (2d Cir. 1994).
20. *RJ Control Consultants, Inc.*, 981 F.3d at 458; *M-I L.L.C. v. Q'Max Sols., Inc.*, No. H-18-1099, 2020 U.S. Dist. LEXIS 140419, at *18-22 (S.D. Tex. Aug. 6, 2020).
21. *M-I L.L.C.*, 2020 U.S. Dist. LEXIS 140419, at *16-17.
22. *Mazer v. Stein*, 347 U.S. 201 (1954).
23. *Situation Mgmt. Sys. v. ASP Consulting LLC*, 560 F.3d 53, 61 (1st Cir. 2009) ("[O]thers may freely describe [the training manual's] process or system by using their own original expression. But others may not appropriate SMS's expression when describing that process or system. SMS's creative choices in describing those processes and systems, including the works' overall arrangement and structure, are subject to copyright protection."); *Feist*, 499 U.S. at 350-51 (finding copyright protection for an original selection or arrangement of underlying unprotectable facts).
24. *Vendavo, Inc. v. Price f(x) AG*, No. 17-cv-06930-RS (KAW), 2019 U.S. Dist. LEXIS 76387, at *15 (N.D. Cal. May 6, 2019); *Datacarrier S.A. v. Woccu Servs. Grp.*, No. 16-cv-122-jdp, 2018 U.S. Dist. LEXIS 50299, at *9 (W.D. Wis. Mar. 27, 2018).
25. *Comput. Assocs. Int'l Inc.*, 982 F.2d at 702 (quoting 3 Nimmer, § 13.03[A], at 13-24).
26. *MDY Indus., LLC v. Blizzard Entm't, Inc.*, 629 F.3d 928, 942-43 (9th Cir. 2010); *Ticketmaster LLC v. Prestige Entm't W., Inc.*, 315 F. Supp. 1147, 1160 (C.D. Cal. 2018); *El-Sedfy v. WhatsApp Inc.*, 2016 U.S. Dist. LEXIS 150760, at *14 (N.D. Cal. Oct. 31, 2016).
27. *Paycom Payroll, LLC v. Richison*, 758 F.3d 1198, 1205 (10th Cir. 2014); *Gates Rubber Co. v. Bando Chem. Indus.*, 9 F.3d 823, 835 (10th Cir. 1993).
28. *Infodeli v. W. Robidoux*, No. 4:15-CV-00364-BCW, 2017 U.S. Dist. LEXIS 228984, at *23 (W.D. Mo. Dec. 18, 2017); *Lilith Games (Shanghai) Co. v. uCool, Inc.*, No. 15-CV-01267-SC, 2015 U.S. Dist. LEXIS 128619, at *15 (N.D. Cal. Sept. 23, 2015).
29. *Productivity Software Int'l, Inc. v. Healthcare Techs., Inc.*, No. 93 Civ. 6949 (RPP), 1995 U.S. Dist. LEXIS 10381, at *9 (S.D.N.Y. July 24, 1995) (citing *Eng'g Dynamics, Inc.*, 26 F.3d at 1344).
30. *Wireless TV Studios, Inc. v. Digital Dispatch Sys.*, No. 07 CV 5103 (RJD)(RER), 2008 U.S. Dist. LEXIS 47620, at *4-5 (E.D.N.Y. June 19, 2008).
31. *SAS Inst. Inc. v. World Programming Ltd.*, No. 2:18-CV-00295-JRG, 2020 U.S. Dist. LEXIS 198116,

at *11-13 (E.D. Tex. Oct 26, 2020); *M-I L.L.C.*, 2020 U.S. Dist. LEXIS 140419, at *9-10; *e-STEPS, LLC v. Ams. Leading Fin., LLC*, No. 19-1637CCC, 2019 U.S. Dist. LEXIS 173062, at *5-11 (D.P.R. Sept. 25, 2019).

32. *Feist*, 499 U.S. at 350-51.
33. *M-I L.L.C.*, 2020 U.S. Dist. LEXIS 140419, at *19-20.
34. *Id.* at *21-23 (citing *Matthew Bender & Co., Inc. v. West Publishing Co.*, 158 F.3d 674, 682-83 (2d Cir. 1998).
35. *Id.*
36. *Datacarrier S.A.*, 2018 U.S. Dist. LEXIS 50299, at *9, *24-25.
37. *Google LLC v. Oracle Am., Inc.*, No. 18-956, 593 U.S. ____ (2021).
38. *Id.* at 1.
39. *Id.* at 15 (emphasis added).
40. Transcript of Oral Argument, *Google LLC v. Oracle Am. Inc.*, No. 18-956 (Oct. 7, 2020) at 4.
41. *Id.* at 18.
42. *Id.* at 55.
43. Respondent Oracle's Brief in Opposition, *Google LLC v. Oracle Am. Inc.*, No. 18-956 (Mar. 27, 2019) at 21-22.
44. *Google LLC v. Oracle Am., Inc.*, No. 18–956, 593 U.S. ____ (2021) (Thomas, J., dissenting) at 1, 4.
45. *Id.* at 4.
46. *Id.* at 6-7.
47. *Id.* at 5-7.