1999

How Can Whelan v. Jaslow and Lotus v. Borland Both be Right? Re-Examining the Economics of Computer Software Reuse

Michael Risch, *Villanova University School of Law*

HOW CAN *WHELAN v. JASLOW* AND *LOTUS v. BORLAND* BOTH BE RIGHT? REEXAMINING THE ECONOMICS OF COMPUTER SOFTWARE REUSE

by MICHAEL RISCH†

ABSTRACT

The various circuit courts of appeal have been unable to agree on the appropriate method of determining when one computer program infringes the copyright in another computer program. This article traces the differences among the circuits, proposes a model to explain what courts are doing, asserts a set of factors that simplify the analysis of determining copyright infringement, and tests those factors against seemingly irreconcilable cases. Finally, the article applies the analysis to unresolved computer software issues of today in order to predict likely outcomes.

I. INTRODUCTION

The basic economic goal of copyright law is to balance an author's incentive to create with his or her ability to build on prior work in order to maximize social wealth. This balance is extremely important for computer software. On the one hand, software is often expensive to create and companies therefore need protection in order to recoup their investment. On the other hand, software is often expensive to create and companies can save costs by reusing pre-existing work.[1] Quite often, the same companies that want to protect their software also want to use pre-

† Michael Risch is an associate with Russo & Hale, L.L.P., Palo Alto, CA. J.D., The University of Chicago (high honors); A.B., Stanford University. Mr. Risch practices intellectual property law, including civil litigation, intellectual property, computer law and licensing. The author wishes to thank Professors Kenneth Dam and William Landes at The University of Chicago for their invaluable comments during the preparation of this article. Email address: mrisch@computerlaw.com.

1. Peter Coffee, *Don't Measure Reliability by Costs*, PC WEEK, Oct 21, 1996, at 24 ("Technologies of software reusability can only get you ahead for a little while: in the long run, they're what you need just to stay competitive).

existing work.[2]

As a result of the competing costs and benefits of copyright protection and the general complexity of computer software, the state of copyright law with respect to computer software has been in flux since the early 1980's and is still not settled. As discussed in detail below, it appears that federal courts have been able to determine efficient economic outcomes based on the cases before them, but they have been unable to settle on a rule that definitively determines how much reuse to allow in each case.

How should a court go about deciding when reuse should and should not be allowed? While case law and traditional economic analysis provide general solutions, they do not guide courts in specific cases. This paper analyzes the problem that courts face in computer software copyright cases, presents an economic model that describes this problem, and presents a set of economic based factors that can be used by courts to obtain efficient solutions in each case.

II. TERMINOLOGY

The following is a discussion of the key copyright, computer, and economic terminology as used in this paper.

A. Copyright Terminology

Copyright *doctrine* is the set of general legal principles that courts use to decide all copyright cases. An example of a copyright doctrinal tool is the *Feist* facts doctrine,[3] which holds that "pure facts" are not copyrightable.

Copyright *rules* are applied to specific types of copyrighted works by courts. For example, courts apply the "forms" rule,[4] which states that blank forms are not copyrightable unless they convey information. This rule is a specific application of several doctrinal tools, including the idea/expression dichotomy[5] and the originality requirement.[6] Rules may be methods of analysis as well. Admittedly, there may be little difference between doctrine and rules in many cases, but the distinction becomes important in computer software cases because different circuits are us-

2. William M. Landes and Richard A. Posner, *An Economic Analysis of Copyright Law*, 18 J. Legal Stud. 325, 348 (1989).

3. Feist Publications, Inc. v. Rural Tel. Serv., 499 U.S. 340, 349 (1991).

4. *See e.g.*, Bibbero Sys., Inc. v Colwell Sys., Inc., 893 F.2d 1104, 1106-07 (9th Cir. 1990).

5. Under 17 U.S.C. §102(b) (1980), expression is copyrightable, but ideas are not. Unless otherwise noted, all code references are to the Unites States Copyright Act, 17 U.S.C., as amended.

6. Under §102(a), a work must be original to be copyrighted. Under *Feist*, the level of originality required is minimal. *Feist*, 499 U.S. at 345.

ing the same doctrinal tools to create differing rules. Current legal rules specific to computer software are inefficient, but doctrinal tools can be used to achieve efficient solutions.[7]

B. Computer Terminology

A program *element* is any part of a program that can be reused. This may include the entire program or a tiny portion of the program.

A computer program's *source code* is the text that a computer programmer writes to create a program.[8] A computer program's *object code* is what the end user typically thinks of as the program; it is a "compiled" form source code that actually manipulates the computer's microprocessor, random access memory, and hard disk.

A program's *user interface* is the portion of the program that the user sees and otherwise interacts with. The user interface includes both *explicit* elements (such as windows, icons, and graphics) and *implicit* elements (such as structure, sequence, and organization).

The *literal* elements of a program are its source code and object code. A program's *non-literal* elements are essentially every other part of the program, with the possible exception of certain parts of explicit user interface. For example, a picture of a person in the user interface would be a literal element. However, a certain type or style of window (such as a grid in a spreadsheet program) would be a non-literal element. The line between literal and non-literal elements in the explicit user interface is often difficult to define, and makes decision-making based on that criteria complicated and fact intensive.

Computer software *reuse* is simply the incorporation of a preexisting element, whether literal or non-literal, in a new program. Reuse includes outright copying of part or all of a program, but also includes use of similar structure, sequence, and organization. Reuse is a more neutral and less pejorative term than "copying" and is favored in the discussion here.

C. Economics Terminology

Transaction costs are costs involved among parties transacting or not transacting, as the case may be. A typical transactions cost in copyright is the cost of negotiating licenses to use another party's copyrighted work. *Administrative costs* are the costs incurred by society in running a

7. Richard Armstrong Beutel, *Software Engineering Practices and the Idea/Expression Dichotomy: Can Structured Design Methodologies Define the Scope of Software Copyright?*, 32 Jurimetrics J. 1, 32 (1991)(notes that regardless of rule methodology in a particular case, well recognized copyright doctrine is used to determine the level of computer copyright protection.)

8. A classic example of BASIC language source code is: PRINT "Hello World."

copyright system. This includes court costs and legal fees. Administrative costs are a specialized form of transaction costs.

An economic *incentive* makes a person more likely to take a certain action to improve his or her *utility*. In general, economic incentives mean that a person will select the alternative that makes the most money. Sometimes this means no action, if the alternative is to lose money. Sometimes this also means actions that increase other forms of happiness that are not monetary based. Those who make laws usually attempt to provide individuals with incentives to make efficient decisions. While economic incentives arguably do not apply to all copyrighted works (such as fine art or unpublished poetry), they certainly apply to computer software.

An *efficient* decision is a decision that maximizes societal wealth. There are two ways to look at efficiency. The first is *unconstrained* efficiency, which means that a decision can be implemented in such a way to maximize wealth to the exclusion of all other solutions. With unconstrained efficiency, it is usually assumed that the parties involved will agree to redistribute assets among themselves,[9] so that decisions to maximize wealth are made without regard to who gets what; the parties will decide how to divide assets among themselves in order to maximize wealth.

Constrained efficiency occurs where one or more factors limit the ability of the parties to reach unconstrained efficiency. A typical constraint in copyright is high transaction costs. *Unconstrained* efficiency might dictate that copyright owners agree to license their works to others in exchange for a royalty. If transactions costs are higher than expected royalties, however, the copyright owner will not have an incentive to enter into a license, and constrained efficiency will result: given high transactions costs, it is more efficient to not license the software.

Economic-based solutions, therefore, have two goals. The first is to reduce the number of constraints that limit unconstrained efficiency. The second is to maximize constrained efficiency given the constraints that could not be removed.

III. THE PROBLEM

Courts have, to date, struggled with the proper way to handle copyright infringement suits relating to computer software. None of the circuits appears to apply the exact same rule.[10] The cases can be divided into two categories: bright line rules that are overreaching and decision

9. *See generally* RICHARD A. POSNER, ECONOMIC ANALYSIS OF LAW §1.1 and §1.2 (Little, Brown & Co. 1992).

10. *But see generally* Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1 (1995).

methodology rules that do not provide enough guidance to lower courts or authors.[11]

The leading case is *Computer Associates International, Inc. v. Altai, Inc.*,[12] in which the Second Circuit determined whether to impose copyright infringement liability using the "abstraction-filtration-comparison" method. Using this method, a court first determines the level of "abstraction" to analyze. The appropriate level may be source code, object code, structure and sequence, individual user interface elements, or total "look and feel" of the program.[13] The court then removes—"filters"—from consideration the non-copyrightable elements of the first computer program. Finally, the court compares the remaining copyrighted elements with the allegedly infringing program.[14]

There are several problems with the *Computer Associates* rule. First, it does not follow universally accepted copyright doctrine that states that a work as a whole must be compared before the filtration of non-protected elements.[15] Filtration was created to determine the amount of the infringement; copyrightability of the whole work should be considered before filtration because of the value created by putting unprotected elements together in an original way.[16]

The second problem is that courts applying *Computer Associates* still do not know how much reuse of each of the program element to allow. As Judge Easterbrook points out in *Nash v. Columbia Broadcasting System*, "the 'abstractions test' is not a test at all . . . [i]t does little to help resolve

11. *See, e.g.*, Linda Skon, Note, *Copyright Protection of Computer User Interfaces: "Creative Ferment" in the Courts*, 27 ARIZ. ST. L.J. 1063, 1082 (1995) (recent court decisions in *Lotus v. Borland* and *Engineering Dynamics v Structural Software* do not provide guidance in determining which aspects of a computer program are copyrightable); John T. Soma, et. al., *Software Interoperability and Reverse Engineering*, 20 RUTGERS COMPUTER & TECH. L.J. 189, 254 (1994) (concluding that reverse engineering case law regarding software interoperability is uncertain).

12. Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

13. *See* Jack Russo and Jamie Nafziger, *Software "Look and Feel" Protection in the 1990s*, 15 HASTINGS COMM. & ENT. L.J. 571 (1993), (discussing the definition and history of "look and feel" protection).

14. *Computer Assocs.*, 982 F.2d at 706-11.

15. *See* Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54-56 (2d Cir. 1936) (L. Hand, J.); CCC Info. Serv. v. Maclean Hunter Market Reports, 44 F.3d 61, 72-73 (2d Cir. 1994) (holding that entire compilation of facts is protectable). Note that, as discussed below, other circuits have corrected this deficiency.

16. *Id. See also* Atari Games Corp v Oman, 979 F.2d 242, 245-46 (D.C. Cir. 1992) ("Breakout" game composed of geometric shapes may be protected by copyright); Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUD. 321, 341 (1995); Anthony L. Clapes, *Confessions of an Amicus Curiae: Technophobia, Law and Creativity in the Digital Arts*, 19 U. DAYTON L. REV. 903, 970 (1994).

a given case . . . ."[17] *Computer Associates* does not offer significant guidance on what features should actually be filtered out in any given case. Courts don't even agree whether the test should apply to literal elements, non-literal elements, or both.[18]

The third problem is that, while the filtering rules are not clearly defined, they are generally less protective of software, and therefore may not provide enough incentive to create.[19]

Other circuits have applied or rejected the *Computer Associates* test with regard to different parts of a program, and have disagreed as to which parts should be subject to the test.

A. Overreaching Decisions: Courts That Do Not Follow *Computer Associates*

In general, courts that have not followed *Computer Associates* have propounded over-broad rules that have been criticized by others. The decisions make bright line rules that cannot possibly cover the myriad of computer software cases that come before courts.[20]

In *Whelan Associates, Inc. v Jaslow Dental Laboratory Inc.*, which predates *Computer Associates*, the Third Circuit held that the only unprotected "idea" of a computer program is the program's purpose, and that everything else is protectable expression as a whole.[21] *Whelan* set a precedent of broad protection that was originally widely approved by courts and has since been criticized thoroughly by many courts and commentators.[22] Despite criticism of the announced rule, it is clear that the *Whelan* court was attempting to achieve a balance between the incentives given to software authors and the ability to create in the future.[23]

In *Lotus Development Corp. v. Borland International, Inc.*, the First Circuit rejected *Computer Associates*, indicating that the Lotus menu structure as a whole was uncopyrightable as a system of operation similar to buttons on a VCR, and that *Computer Associates* should not be extended to apply to literal elements such as menus and screen dis-

17. Nash v. Columbia Broad. Sys., 899 F.2d 1537, 1540 (7th Cir. 1990).

18. Bateman v. Mnemonics, Inc., 79 F.3d 1532,1545 (11th Cir. 1996).

19. Brett A. Carlson, *On the Wrong Track: A Response to the Manifesto and a Critique of Sui Generis Software Protection*, 37 Jurimetrics J. 187, 191 n.33 (1997).

20. Dennis M. Carleton, Note, *Lotus Development v. Borland International: Determining Software Copyright Infringement is Not as Easy as 1-2-3*, 56 U. Pitt. L. Rev. 919, 945 (1995) (noting that the test by district court in *Lotus v. Borland* is flawed as compared to "abstraction-filtration-comparison" test).

21. Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986).

22. Jack E. Brown, *"Analytical Dissection" of Copyrighted Computer Software—Complicating the Simple and Confounding the Complex*, 25 Ariz. St. L.J. 801, 814 (1993).

23. *Whelan*, 797 F.2d at 1235.

plays.[24] In *Plains Cotton Cooperative Ass'n v. Goodpasture Computer Service, Inc.*,[25] the Fifth Circuit held that the nature of the cotton industry created the need for similar programs. The *Plains Cotton* decision explicitly rejects *Whelan*, and provides a test that looks at the work as a whole as well as industry requirements, which is different from the *Computer Associates* test.

The overreaching problem is not confined to computer software cases. In *BellSouth Advertising and Publishing Corp. v. Donnelley Information Publishing, Inc.*,[26] a telephone yellow pages copying case, the court made an arguably efficient decision by allowing some copying of business name and telephone information from the plaintiff's yellow pages. At the same time, however, the court stated: "Ultimately, the district court erred by extending copyright protection to the collection of facts . . . based on the uncopyrightable formative acts used to generate those listings."[27] Under the stated rule, virtually no yellow pages information could be protected because all collections of facts would be uncopyrightable, even though the selection of which facts to find, as well as which facts to print, might be original and protectable as a whole. The court later diluted its statement, but still leaves the impression that original selection of facts is not enough to protect against infringement.[28] Such a result cannot possibly be what the court intended. Even the Supreme Court in *Feist*[29] indicated that a telephone book with creatively selected but unoriginal underlying facts could be protected as a whole.

B. Not Enough Guidance: Courts That Modify *Computer Associates*

Some courts have rejected bright line rules in favor of a modified *Computer Associates* rule. These courts disagree about when and how to apply the "abstraction-filtration-comparison" method. Regardless of the rule applied, however, lower courts and software authors following these decisions lack guidance about how to decide any given case.

In *Gates Rubber Co. v. Bando Chemical Industries, Ltd.*, the Tenth Circuit expanded *Computer Associates* by adding what has been called an initial holistic comparison.[30] The court reasoned that examination of

24. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 814-15 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

25. Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1262 (5th Cir. 1987).

26. BellSouth Advertising and Publ'g Corp. v. Donnelley Info. Publ'g, Inc., 999 F.2d 1436, 1445 (11th Cir. 1993).

27. *Id.* at 1441.

28. *Id.* at 1445.

29. Feist Publications, Inc. v Rural Tel. Serv., 499 U.S. 340, 349, 362 (1991).

30. Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 832-33 (10th Cir. 1993).

both programs in their entirety, prior to filtration, would reveal any patterns of copying.[31] The *Gates* court noted that *Whelan* is too broad in holding that a program can only have one idea, but is otherwise sound as to its holding that the structure and sequence of a program may be copyrighted.[32] This appears to be contrary to *Computer Associates*.

In *Engineering Dynamics, Inc. v. Structural Software, Inc.*, the Fifth Circuit adopted *Computer Associates* as expanded by *Gates,* and attempted to reconcile its application of *Computer Associates* and *Gates* with its earlier *Plains Cotton* decision.[33]

In *Apple Computer, Inc. v. Microsoft Corp.*, the Ninth Circuit essentially adopted the *Computer Associates* methodology, but added that a compilation of uncopyrightable elements could be protected from wholesale or virtually identical copying.[34] Under this rule, the court performs "analytical dissection" to determine which program elements are protected. After the dissection, the court grants either broad (if most elements are protectable) or narrow (if most elements are unprotectable) protection to the work as a whole, and compares the two works as a whole.

In *Bateman v. Mnemonics, Inc.*, the Eleventh Circuit adopted yet another form of the *Computer Associates / Gates* test.[35] However, just a few months later, in *MiTek Holdings, Inc. v. Arce Engineering Co.*, the court appears to have applied instead the threshold test from *Lotus v. Borland* to determine whether the single reused element was copyrightable, but at the same time appears to have criticized the rule promulgated in *Lotus*.[36] Despite apparently applying a threshold test and finding the element uncopyrightable, the *MiTek* court also applied the methodology from *Apple* by comparing the works as a whole even though only one element was allegedly copied.[37] Finally, the court applied the *Bateman* test to the programs after uncopyrightable elements, including the primary element at issue, had been removed.[38] The Eleventh Circuit probably understands its methodology perfectly well, and it may in fact be the best methodology available given the procedural posture of that specific case, but parties reading *MiTek* will have a difficult time deci-

31. *Id.*

32. *Id.* at 836, 840.

33. Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1342 (5th Cir. 1994).

34. Apple Computer Inc. v. Microsoft Corp., 35 F.3d 1435, 1442 (9th Cir. 1994)(noting that the court's methodology was similar to the *Computer Associates* rule, not an adoption of it).

35. Bateman v. Mnemonics, 79 F.3d 1532, 1544-45 (11th Cir. 1996).

36. MiTek Holdings, Inc. v. Arce Engineering Co., 89 F.3d 1548, 1557 (11th Cir. 1996).

37. *Id.* at 1558-59.

38. *Id.*

phering it. This is an example of how imprecise language in computer copyright cases can create confusion and conceivably lead to a misreading of what the court's opinion really means.

Some guidance beyond the methodology suggested by these cases is important for courts and software authors. This is because many cases are decided on summary judgment or after a bench trial. Further, juries can be expected to make better decisions if they are given more concrete guidelines.

C. Consequences of the Problem

Courts have been unable to create generalized computer software rules that lead to an optimal amount of reuse in each case. While bright line cases such as *Whelan* and *Lotus v. Borland* arguably allow for predictable results, the rules lead to inefficient outcomes and courts have not blindly applied them. Lower courts are caught between a rock and a hard place.[39] They must either follow the rule and make an inefficient decision, or they must distinguish appellate decisions from most cases at bar.

If every case must be distinguished, however, computer software companies will not be able to make decisions for the future. The maxim that "hard cases make bad law" is proven here, because all cases are "hard" and consequently circuits disagree on the proper rule.[40] As a consequence, the classification a court selects for a particular computer program may reflect the case's result rather than deliberative reasoning.[41]

Uncertainty and instability in the law creates problems for software companies deciding to invest in new products. Intellectual property is similar to other property,[42] and poorly defined property rights have several disadvantages.

First, clearly defined property rights give the parties an incentive to contract for use of assets, if necessary.[43] It would be wasteful for companies to negotiate for use of software elements that may be freely reused without a license.

Second, without clear property rights, the number and cost of dis-

39. *See, e.g.*, William F. Patry, *Copyright and Computer Programs: It's All in the Definition*, 14 Cardozo Arts & Ent. L.J. 1, 4 (1996) (arguing that courts have "overreacted" and propounded inconsistent definitions of "originality" and "computer program" thus making consistent and efficient decision making difficult).

40. Brown, *supra* note 22, at 804 (the number and types of rules applied by courts to computer software is extremely complex and error-prone).

41. *Id.*

42. Frank H. Easterbrook, *Cyberspace and the Law of the Horse*, 1996 U. Chi. Legal F. 207, 207-08 (1996).

43. Dam, *supra* note 16, at 354, 365 (gains from contract require that a follow-on firm not be able to copy outright); Posner, *supra* note 9, at §3.1.

putes increases,[44] with the potential side effect of running small competitors out of the market.[45] This is not always true because extremely high litigation costs may force competitors to contract with each other even if they need not under the law. One way or the other, however, the effect is an inefficient market.

Third, clearly defined property rights allow software developers to select an efficient amount of investment. If companies do not know how much protection their software will receive, they will not be able to decide how much to spend creating the product or which measures of protection to use.[46] Further, uncertainty may create a "chilling effect" which causes authors to reuse less than they might be allowed by law due to a fear of lawsuit.[47]

This paper proposes a solution to make each case less difficult to decide in the future and to give authors guidance before going to court.

IV. THE ECONOMIC MODEL

The economic model proposed here differs slightly from the traditional analysis in the literature, but more accurately explains what courts have been doing.

A. The Traditional Analysis

Copyright creates societal benefits by giving authors an incentive to create original works. At the same time, copyright makes it more difficult to create new works because the later works might be considered an infringing copy of earlier works. Societal wealth is maximized by weighing incentives to create against access costs at varying levels of copyright protection.[48] Landes and Posner suggest that the very same companies that want copyright protection also want to use prior works as building

44. Dam, *supra* note 16, at 364 (uncertainty means that litigation would be required to resolve disputes); Jonathan E. Retsky, *Computer Software Protection in 1996: A Practitioner's Nightmare*, 29 J. Marshall L. Rev. 853, 854 (1996) (no uniform precedent makes copyright practice unpredictable).

45. Clapes, *supra* note 16, at 963 (summary judgement is extremely important to small software companies).

46. There are a number of protection alternatives other than copyright. *See* Landes and Posner, *supra* note 2, at 329-31.

47. Matthew P. Larvick, Note, *Questioning the Necessity of Copyright Protection for Software Interfaces*, 1994 U. Ill. L. Rev. 187, 203 (1994).

48. Landes and Posner, *supra* note 2, at 342-43. *But see* Glynn S. Lunney, Jr., *Reexamining Copyright's Incentives-Access Paradigm*, 49 Vand. L. Rev. 483, 492-93 (1996), who points out that broad protection of copyright diverts creative resources from other productive work by creating incentives for "over-creativity," and copyright should therefore be limited regardless of the need for access. It is not clear how this argument would apply to works with more functional characteristics, such as computer software.

blocks for future works.[49]

The traditional economic analysis is *ex ante* based. That is, the parties make decisions about the appropriate level of protection at some early point in time prior to action. *Ex ante* analysis is necessarily based on averages, or *expected value*[50] of benefits and costs; the level of copyright protection and allowed reuse is determined so that it will maximize social benefits in the broad range of cases, even if some specific cases do not maximize wealth. The administrative costs of deciding each case outweigh the costs of a few inefficient applications of the rule.

After an efficient level of reuse is determined, the parties then decide how much software to create and what to reuse. They also decide what they should reuse freely and what they should license from others.

Because the level of allowed reuse creates incentives to make efficient decisions, the decisions made by individual authors maximize social wealth. A key empirical question is exactly how much benefit society gains by the reuse decisions of authors.[51] Theoretically, this could be measured through data gathering. Practically, however, courts must make this determination using the evidence in the cases before them and without any extensive cost and benefit research. Therefore, appellate courts and commentators have set forth general rules based on the economic analysis that are intended to guide lower courts in their decision making. These same general rules are intended to grant the efficient level of protection and thus induce efficient decisions by authors.

1. *Problems with Traditional Analysis: Misguided Proposals*

As a result of the traditional economic analysis, many commentators incorrectly suggest eliminating protection as soon as the copyright owner recoups his or her costs.[52] This suggestion does not comport with either

---

49. *But see* Pamela Samuelson and Robert J. Glushko, *Comparing the Views of Lawyer and User Interface Designers on the Software Copyright "Look and Feel" Lawsuits*, 30 JURIMETRICS J. 121, 125-26 (1989), who conclude that 79% of software designers oppose broad "look and feel" protection for software. The problem with this survey is that the software designers are not paying their own salaries to create software nor do they see all of the benefits of protection. Thus, on the whole, they prefer to reuse more often. The parties bearing the costs and realizing the benefits may respond to such a survey quite differently.

50. Expected value is essentially the sum of future possibilities weighed by the probability that each possibility will occur. More technically, the formula for expected value E(x) is $\int_y^z f(x)xdx$ where x is a possible outcome, f(x) is the probability of that outcome, and x falls in [y,z], the range of possible outcomes.

51. *See, e.g.*, Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 STAN. L. REV. 255, 264-65, 277-84 (1997) (examining costs and benefits of reuse and arguing that limited copyright and patent protection is best way to encourage reuse).

52. *See, e.g.*, Barron Yanaga, Note, *An Economic Analysis of Computer Software Copyright: A Welfare Model of Intellectual Property Rights*, 11 COMPUTER/L.J. 173, 179 (1991)

the traditional or the proposed economic analysis. The problem is that companies do not know whether a product will be successful or not prior to creating the computer program. Thus, authors base their decisions on the expected value of future revenues.

If protection were eliminated as soon as revenues outweighed costs, then companies would not create software because the truncated expected value of revenues might never exceed expected costs.[53] Essentially, profits would be capped at breakeven or just above, yet could—in many states of the world—be negative. A second problem with this analysis is that courts have never limited copyright in any work simply because the author earned a large amount of money.[54] Finally, it is an unrealistic solution because copyright laws as applied are the norm, and it is unlikely that such a limited application would ever be used.[55]

The only solution for misguided proposals is further discussion, analysis, and avoidance of such proposals.

2. *Problems with Traditional Analysis: Unknown Variables*

As the discussion below shows, the courts noted above each made an efficient decision yet propounded inefficient rules. There are several possible reasons for this "do as we do, not as we say" message sent by the circuit courts. At bottom, however, is the notion that due to the broad range of facts that comes before courts, the administrative cost of finding the exact and most efficient general rule to propound is higher than the administrative cost of making an efficient decision in a single given case. While courts attempt to make efficient decisions generally, they are unable to consider (or *internalize*) all of the social costs of their opinions, especially when they are given a fixed set of facts in a given case.[56]

("The goal of software copyright law, therefore, is to award the innovator a right that is equal to the software's marginal value to society.").

53. If w equals expected value, then courts would allow complete copying when w is earned. Some amount above w, called u, would be earned, but u would be small due to nearly costless competition from the low marginal cost of creating copies. Under this rule E' (x) would equal $\int_y^{w+u} f(x)xdx$ which is less than E(x) in all cases.

54. Brown, *supra* note 22, at 836 (stating that commercial success should not bar copyrightability).

55. 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, 2562 (1994) (arguing that copyright is the norm, so it makes little sense to propose sui generis protection); Greg S. Weber, *The New Medium of Expression: Introducing Virtual Reality and Anticipating Copyright Issues*, 12 COMPUTER/L.J. 175, 190 (1993) (noting that fact/expression dichotomy unlikely to change).

56. *But see* Mark A. Lemley, 10 *supra* note 10, at 19 ("To some extent, the differing outcomes courts have reached in applying the filtration test may simply reflect their political predispositions towards 'high' or 'low' protectionism.").

Problems with *ex ante* analysis have been noted before in other areas of law. Oliver Williamson notes:

> Although it is instructive and a great analytical convenience to assume that agents have the capacity to engage in comprehensive *ex ante* contracting (with or without private information) the condition of bounded rationality precludes this .... Accordingly, the *ex post* side of a contract takes on special economic importance. The study of structures that facilitate gap filling, dispute settlement, adaptation and the like thus become part of the problem of economic organization. Whereas such institutions play a central role in the transaction costs economics scheme of things, they are ignored (indeed, suppressed) by the fiction of comprehensive *ex ante* contracting.[57]

Likewise, Easterbrook and Fischel state:

> Yet it is costly for the parties . . . to ponder unusual situations and dicker for the adoption of terms .... Court systems have a comparative advantage in supplying answers to questions that do not occur in time to be resolved *ex ante*. Common law systems need not answer questions unless they occur. This is an economizing device; it avoids working through problems that do not arise. The accumulation of cases dealing with unusual problems then supplies a level of detail that is costly to duplicate through private bargaining.[58]

In copyright law, and especially in computer software, copyright authors cannot know whether their particular use of a prior work will constitute illicit copying. Further, the state of copyright law, as noted above, does not give enough guidance to yield decisive *ex ante* conclusions because an appropriate "level of detail" is not present in current circuit decisions. The economic model and decision factors proposed in this paper seek to unleash the economic power of *ex post* analysis envisioned by Williamson and Easterbrook and Fischel, while avoiding the costs of detailed case by case determination.

B. The Proposed Economic Model

The proposed economic model is relatively simple, and is based on a theory that courts maximize the change in net social benefits in the cases before them.[59] The change in social benefits is calculated from the facts as they stand just prior to reuse. While courts under the proposed model also consider future behavior, the decisions courts make are based on the

57. Oliver E. Willamson, Transaction Cost Economics, *reprinted in* Foundations of Corporate Law 12, 13 (Roberta Romano ed. 1993).

58. Frank H. Easterbrook and Daniel R. Fischel, The Structure of Corporation Laws: The Corporate Contract, *reprinted in* Foundations of Corporate Law 101, 108 (Roberta Romano ed. 1993).

59. Ginsburg, *supra* note 55, at 2559 (stating that software industry is thriving, so current copyright law must be doing something well).

facts before them in addition to the potential future behavior.[60]

The model is similar to a game theory problem of software creation. Because the relevant facts are determined after creation of the first program, but before reuse in the second program, courts will maximize social benefits at the time of the decision to reuse. Thus, the first author will make an investment decision based on an estimate or guess about what others will do in various states of the world. Then, the second author will make a decision about reuse based on the likely outcome based on the state of the world at the time of the decision to reuse. The goal of the court is to encourage new works at both periods of time, and if social benefits are maximized only before the first work is created, this dual maximization may not occur because of unknown facts at that time of creation.

1. *Differences From Ex Ante Analysis*

Perhaps the best way to describe the model presented here is to describe how it differs from traditional *ex ante* analysis. In general, courts seeking to maximize the change in social wealth based on facts as the reuser sees them, or *ex post*, may come to different decisions than courts trying to maximize wealth generally with some predefined, or *ex ante*, rule that is assumed to affect the behavior of all parties in the past, present, and future.[61]

The intuitive reason for this difference is that lawmakers and economic analysts examine behavior prior to making or proposing laws, while courts must interpret and make rules with a specific case in mind. A more rigorous explanation for the difference is that there are some facts that cannot be estimated beforehand.[62] Because of this, determin-

60. For an interesting example, *see Advanced Micro Devices, Inc. v. Intel Corp.*, 9 Cal. 4th 362, 370-71 (1994), where the California Supreme Court upheld an arbitrator imposed license to copyrights and patents in an arbitration relating to breach of contract in order to stop current and future disputes between the parties which were not before the arbitrator.

61. *See, e.g.*, Landes and Posner, *supra* note 2, at 341-43 (optimizing *ex ante* welfare based on *ex ante* level of protection).

62. Symbolically, the expected social benefits under the traditional model are:
$\iiiint f(v, w, x, y, z)B(v, w, x, y, z)dvdwdxdydz$ where v, w, x, and y are facts, z is the level of copyright protection, f is the density/probability function of each combination of facts and levels of protection, and B is the level of social benefits for each set of facts and the level of copyright protection. Under this model, the court sets the level of copyright protection, and the parties can select or estimate the facts that maximize expected benefits based on the level of protection.
On the other hand, expected social benefits under the proposed model are:
$\int f(z \mid v, w, x, y)B(z \mid v, w, x, y)dz$ where the facts (v,w,x,y) are given, and the court can only optimize expected social benefits by changing the level of copyright protection in the given case. Under this model, the parties have already chosen the facts, and the court can only pick an optimal outcome by varying the level of protection. To a lesser extent, the court can set a level of protection that will encourage an efficient selection of facts in the future.

ing the expected values in an *ex ante* calculation does not yield optimal results in many cases before courts.

An example from tort law is the determination of negligence. In general, no negligence will be found where the defendant could not have learned certain facts without expending costs that exceed the costs of the accident. Thus, the court is left to determine after the fact, or *ex post*, whether the costs of learning the information exceed the costs of the accident. While *ex ante* analysis creates the general rule, the court must *ex post* determine whether the defendant wasted societal resources by not learning the necessary facts.

The current lack of clarity in copyright rules promulgated by the various circuits implies that some variables are not estimable beforehand. Therefore, the differences between an *ex ante* and an *ex post* analysis will occur primarily where facts could not reasonably be determined or estimated beforehand, and therefore cannot be used to make efficient rules beforehand. While *ex ante* analysis will provide a broad framework for general rules, a court cannot make a final determination until it reviews the facts of each case. The model should have the following effects on incentives as compared to *ex ante* analysis:

a. To the extent that *ex ante* likelihood of a case outcome can be estimated, the incentives of the first author should not be affected. Whether courts consider social benefits prior to creation or prior to reuse, the first authors will consider *ex ante* expected value because that is all they know at the time of the decision to create.

b. If *ex ante* rules do not present an accurate or estimable view of how reuse will be treated by courts, then incentives to create the first work will change, but should become more efficient if the *ex post* facts can be estimated. This is because estimation of *ex post* facts will be more accurate than if considered *ex ante* due to improved information. Accuracy should lead to efficient decision-making if the law creates incentives for efficiency.

c. If *ex post* facts cannot be estimated, then incentives to create the first work should not change, because the first authors will have no more information than in the *ex ante* analysis.

d. Incentives to reuse may change, but should move in a direction that increases social wealth. Because the reuser can now consider the facts at the time of the decision whether to reuse, the probability that reuse will be allowed can more accurately be determined than if the rule is based on a variety of facts that may or may not be present at the time of the reuse decision, but were expected or guessed at the time of creation of the first work. If the reuser can more accurately determine the likelihood of winning or losing in potential litigation, the reuser will be more likely to make the social benefit maximizing decision at the time of reuse.

In many cases the outcome will be the same under *ex ante* and the proposed *ex post* approach, but the results may differ in some cases, and those differences lead to the economic factors proposed in this article. If the results were always the same, then it is unlikely that the problems with rules promulgated by the circuits would exist at all.

2. *An Example of the Differing Analyses*

An example of the differences between the models may also be helpful in describing how the proposed model operates. In *Gracen v. Bradford Exchange* the plaintiff won a contest by painting a picture that best captured the "essence" of Dorothy from *The Wizard of Oz*.[63] However, the plaintiff and defendant were unable to agree on a contract, so the defendant had an employee "clean up" the painting to publish on a decorative plate.[64] Applying traditional *ex ante* economic analysis, Judge Posner reasoned that the administrative cost of determining infringement was high in cases where a new work closely reuses a large portion of a pre-existing work owned by the alleged infringer; here the defendant had rights to Dorothy's image and the plaintiff's work was also based on Dorothy's image. He thus denied copyright protection, requiring a greater showing of originality.[65]

This outcome was efficient *ex ante* because the author would in general receive payment for her work and thus have an incentive to create, even if copyright did not inhere. Further, a small incentive is all that would be necessary because of the small degree of creativity required to re-create the image of Dorothy. In addition, the administrative cost of determining the original elements for copyright protection in derivative works based on live people, photographs, or motion pictures would be high in most cases.

Under the *ex post* analysis, however, a judge would note that in this case the author in fact did not get paid for her work, and thus it would improve the net social benefits to require payment for the reuse; otherwise, people might stop creating new designs that might be used on decorative plates. Also, the administrative costs described above did not even exist in the case at bar, as copying was virtually admitted. Finding infringement would have been efficient *ex post* because it would have created incentives to not breach economic relationships with little offsetting administrative or other social costs.[66]

63. Gracen v. Bradford Exch., 698 F.2d 300 (7th Cir. 1983).

64. *Id.* at 301-02.

65. *Id.* at 304-05.

66. As noted below, breach of an economic relationship is one of the factors that courts should consider to determine the level of protection to grant.

Gracen is an extreme example of *ex ante* analysis making a rule that is ostensibly efficient for all cases in the future, on average, without regard to the current case. The *ex post* model will often lead to conclusions similar to the *ex ante* model, but a focus on the facts at bar may also allow a focus on "unexpected" behavior that might occur.

3. *The Symbolic Function and Its Implications*

The basic symbolic function facing the court is:

$\Delta S = \Delta B - \Delta C$ where S is net social wealth, B is total social benefits, and C is total social costs.

This can be expanded into:

$\Delta S = (\Delta B_p - \Delta C_p) + (\Delta B_r - \Delta C_r) + (\Delta B_s - \Delta C_s)$

where the subscripts *p*, *r*, and *s* represent benefits and costs to producers, software reusers, and the rest of society respectively. The court seeks to maximize ΔS, the change in social wealth at the time of the choice whether to reuse with its decision to allow or disallow reuse.

There are two important implications of this formulation. The first implication is that sometimes courts are unable to increase S by intervention and the parties may not sue because they have nothing to gain. If this is so, then the reuse "market" will be stable and in equilibrium at the time of reuse, and the parties will not litigate. This does not always mean that social benefits are maximized. It instead means that courts do not get the chance to improve things; the status quo is a constrained efficient outcome. If the *ex ante* rules are efficient (either constrained or unconstrained) and people follow them, then litigation would never arise.[67]

However, if a court can improve one party's wealth, someone will sue and the court must perform an *ex post* analysis. This is a form of market failure. If the court finds infringement and does not allow software reuse, then the judge has determined that he or she cannot do any better than what already existed before the market failure.[68] If however, the court finds that allowing reuse will improve net social benefits, then it will allow at least some reuse.[69]

The second implication is that courts do not face a continuous social benefit function as is often assumed in *ex ante* analysis. This means that courts sometimes do not fully maximize the change in social benefits (un-

67. Compare this with the familiar argument that there is no negligence in the world because everyone exercises efficient care knowing that they will have to pay if they do not exercise efficient care.

68. Technically, this follows from the fact that ΔS is roughly equal to the derivative of S. Thus, if the court can do no better than the status quo, then ΔS will equal zero, satisfying the mathematical maximization requirement that the derivative of S equal 0. In other words, anything other than the status quo reduces net social benefits.

69. Technically, this means that the court will allow some reuse whenever $\Delta S > 0$.

constrained efficiency) because they are unable to tailor a precise solution. In other words, courts must sometimes settle on a second best or even a third best solution because their choices are constrained to the facts and parties before them. Rulemakers considering an *ex ante* economic analysis, however, may often assume that these constraints do not exist and therefore may recommend different outcomes than would be actually efficient at trial.

C. Justifying the Model: Market Equilibrium

A good test of whether the model describes decisions made by the courts is whether the model describes or predicts behavior of parties who have chosen not to litigate. If the model describes market equilibrium, then it can be used to describe both market failures and the public's perception of what courts are actually doing. The following are several examples where the parties did not seek a court determination because the court could do no better than the status quo or because the parties could not gain.

1. *Object Oriented Programming*

One example of market equilibrium in computer software reuse is "Object Oriented Programming" ("OOP"). With OOP, a computer programmer can write or use pre-written "objects" that are self-contained units which perform certain programmatic tasks. A recent application of OOP is Microsoft's ActiveX (also known as "OCX") technology. With ActiveX, programmers can execute specific tasks from different computer languages, from World Wide Web pages, or even from word processor macro scripts.[70]

Objects can usually only be reused by wholesale copying; nonetheless, object authors in general neither bring suit nor charge other programmers for reuse when the object is redistributed to many end users so long as the first use of the object is licensed by the programmer. In many cases, object writers do not charge for reuse at all. For example, Borland's Delphi compiler uses "visual components" that are used as elements to create programs. There are a very large number of such components available at no charge or minimal charge with unlimited distribution rights so long as the component is embedded into another program and not resold individually.

The reason court intervention is not generally sought with object oriented programming is that courts cannot improve the economic standing of any individual. Societal benefits are large when new programs are

70. A macro is a mini-program written in a computer application program's special language.

created. Further, costs to new developers are lowered if preexisting objects can be used.[71]

Most importantly, the authors of the objects benefit when others use objects; broad use of objects can ensure further revenues or distribution of some base program. This is an application of positive network externalities[72] that benefit the first author as well as society. A paradigm example of this type of network externality is Microsoft's free distribution of Internet Explorer, its World Wide Web browser. Program users gain increased compatibility with their operating system and increased consistency with web content because developers have an incentive to "develop to" Microsoft's version of the World Wide Web.[73] Microsoft gains through the purchase and use of other Microsoft products, such as Windows95, WindowsNT (including a World Wide Web server), and html (web page) authoring and managing tools. A company using this strategy hopes to "tip" the market in order to remove the competition because no user would want to use the competing programs.[74]

Microsoft's practice has been questioned by the United States Department of Justice as a monopoly practice due to the large gain Microsoft has made in this area in such a short time.[75] A key question in the debate is whether consumers benefit as well as Microsoft, and whether free distribution (or more appropriately, forced distribution) limit competition and thus eliminate innovation.[76]

This type of positive externality also exists for "smaller" objects as well as for third party authors. Continuing the Delphi example, Borland was able to create a large market share by encouraging third party com-

71. *But see* Soma, *supra* note 11, at 193 ("There are no off-the-shelf interchangeable software components that can be used to build, improve or repair a software application."). It appears that the expansion of object oriented technology since 1994 may have proved this statement wrong.

72. *See* Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1066 (1989) ("'network externality' describes a class of goods for which the utility . . . derived from the good's consumption increases with the other persons consuming the good"). *See also* Dam, *supra* note 16, at 345 (arguing that these benefits and costs should not be called "network" externalities because there is no actual connection between machines). Regardless of terminology, these are clearly a social and private benefit under the economic model proposed here.

73. Dam, *supra* note 16, at 352-53 (calls this contractual attachment development).

74. James Boyle, *Intellectual Property Policy Online: A Young Person's Guide*, 10 HARV. J.L. & TECH. 47, 50 (1996) (noting that even pirated programs may help a company's market position).

75. *Id.* (interestingly, the author uses free distribution of Microsoft's primary competitor, Netscape, as his example of this phenomenon only one year earlier).

76. *See* United States v. Microsoft Corp., 980 F. Supp. 537, 540 (D.D.C. 1997) *rev'd* 147 F.3d 935 (D.C. Cir. 1998)(issuing injunction and appointing special master to advise on factual issues). While the injunction was reversed, at the time of publishing Microsoft is at trial on this very issue.

ponent developers to share their components. Component developers had a vested interest in sharing the components because other developers would write new Delphi programs, and sales of Delphi would be sufficient to ensure further support and improvements by Borland.[77] In addition, users of the generalized components gained by not having to recreate those components and by writing standardized applications.[78] Finally, society gains through increased competition and productivity.[79]

At market equilibrium, such as the Delphi case, software companies gain enough to not challenge reuse and threaten positive network externalities. Menell notes, however, that benefits are lost if software companies ignore externalities in order to maximize their own revenue through broad copyright protection.[80] For example, Baird, et. al., argue that Lotus 1-2-3 was such a dominant product that its menu hierarchy created a negative network externality.[81] People had no incentive to use other programs due to the cost of learning a new menu system, and Lotus had no incentive to allow others to use its menu system because people were continuing to buy Lotus 1-2-3. Under the model, therefore, courts would be expected to improve on the status quo and allow reuse where a large potential benefit exists by allowing reuse.

2. *File Translation and Compatibility*

Another example of market equilibrium is file translation and compatibility. There are two types of reuse relating to file translation: filtering and plug-compatibility. With filtering, one computer program will translate or convert the format of a file created in another program to be used with that program. For example, the Corel WordPerfect word processing program includes a filter that will convert a Microsoft Word word processing program file into WordPerfect's default file format.

Sometimes programs read and write in the same file format by default and without translation programs. With plug-compatibility, multiple programs are able to use files created by another program author as their own. For example, Links and Microsoft Golf are two popular computer golfing games. Each of these programs has several different golf course designs that are stored in program files. Each of the programs,

77. Delphi is now about 7 years old and in its fourth revision.

78. *See* Jamie Lewis, *Tool Portability is Key for Future Net App*, PC WEEK, Feb. 3, 1997, at 82 ("Portability of the skill sets and tools in which you invest, then, is just as important as the portability of the applications you write.").

79. *See* Coffee, *supra* note 1, at 24 ("Pretty soon everyone was doing better, but only able to keep up with a new standard of competence. . . . And with more software being reused, more of the developer's work will be design, rather than reverse engineering and defect resolution. . . . The quality of software product will rise. . . .").

80. Menell, *supra* note 72, at 1068-71.

81. BAIRD, ET. AL., GAME THEORY AND THE LAW 212-213 (1994).

however, can read the golf course design files of the other. Therefore, owners of each program can use golf courses designed for either, and course designers can sell to owners of either program. In addition, all World Wide Web browser programs read and interpret HTML[82] format files, although there are certain codes that only certain browsers can interpret. Finally, numerous database programs use the dBase IV (.dbf) formats for their database files.

File translation and compatibility is quite arguably copyright infringement; it is a copying of original expression created by the first author.[83] Despite this, cases relating to file translation and compatibility are almost unheard of.[84] The model explains why parties often litigate regarding executable program compatibility,[85] but not for file translation. The model suggests that in cases of file translation, the user may have large switching costs,[86] due to loss of use of existing files, such that inability to translate files would not allow competitors to enter the market. Without competition, social benefits would be reduced, and the first author can be relatively sure that a court would allow reuse.

Further, the first software authors do not bring suit because they are able to create their own filters for competitive programs and therefore do not have an incentive to eliminate their use.[87] However, in the case of executable program compatibility such as compatible video games, no reciprocal use exists, so companies are more likely to litigate.

3. *Free Software Foundation*

Another market-based solution is the "copyleft" policy of the Free Software Foundation ("FSF").[88] The programmers associated with the FSF believe that all software should be freely available and modifiable, and they make all software created by them available in both object code

82. HyperText Markup Language.

83. *See* MAI Systems Corp. v. Peak Computer, Inc., 991 F.2d 511, 519 (9th Cir. 1993) (holding that loading a program into memory is the creation of a copy).

84. *But see* Engineering Dynamics v. Structural Software, 26 F.3d 1335, 1347 (remanding case to district court to hear evidence on the extent that program input and output formats are determined by market requirement).

85. *See, e.g.,* Atari Games Corp. v. Nintendo of America Inc., 975 F.2d 832 (Fed. Cir. 1992); Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

86. Switching costs are the costs users must expend to switch from one program to another, including training. Where expensive computer hardware is required to run software, the costs of purchasing hardware may be so high relative to the value of switching as to constitute "lock in." Dam, *supra* note 16, at 346-47.

87. *See, e.g.,* Lotus Dev. Corp. v. Paperback Software, Inc., 740 F. Supp. 37, 78 (D. Mass. 1990) (despite suing for infringement, "Lotus itself has written such a capability for translating macros among different-language versions of 1-2-3.").

88. *See* GNU's Not Unix! <http://www.gnu.org> All information discussed here is available at this World Wide Web site.

and source code form. However, because copyright allows protection of derivative works, the FSF copyrights its software and requires that any improvements to the software must be made available in both source code and object code, so that others may use, understand, and modify it.[89] In other words, the FSF uses copyright law to create a market-based solution to achieve its own ends.

This market-based solution is one that requires no court intervention so long as users of FSF code adhere to the license agreement. FSF creates new software, which is a social benefit, without maximized profits as the primary goal. Users of the software are free to modify and improve the software, another social benefit. Because the FSF authors and users have personal incentives to create without monopoly profit, the court can do no better than let the status quo continue.

If, however, a user breached the license agreement, the court would be able to improve societal benefits by finding infringement or specifically enforcing the agreement. If the FSF were unable to enforce its interests through copyright law—for example if a party received the software from a third party and not subject to the FSF's license agreement—its authors might not have the same incentive to create new software and thus social benefits would decline.

An interesting question is whether the availability of free software decreases the incentive for others to create competing software due to lack of demand at any price above zero. This is a viable concern, but once again it would be impossible for courts to improve the situation. Parties are free to enforce copyright law as they wish, and the court could take no action, under current law at least, to improve on the market equilibrium. In addition, the FSF encourages authors to charge whatever price they can get in the market; thus price competition by new entrants is still a possibility under the FSF agreement. As an empirical matter, the availability of free software appears to have done little to affect software innovation. If anything, free software may in fact spur innovation, as discussed below regarding World Wide Web browsers.

V. APPLICATION OF THE MODEL: DECISION FACTORS

While it is important to describe cases generally and discuss current market equilibrium, it is also important to use the model to guide future decisions. The set of decision factors proposed below applies the model in a way that is practically useful.

89. It is extremely rare for a commercial software publisher to distribute source code. Netscape's Navigator and Network Associates Pretty Good Privacy are two of the few exceptions.

A. Why Use Decision Factors

Thus far, the proposed economic model is simply an after the fact descriptive tool and implies that decisions must be made on an *ad hoc* basis. Ad hoc decisions are not unusual given the wide variety of computer software copyright cases and the number of different doctrinal tools that might apply in any given case.[90] As the Supreme Court notes in *Campbell v. Acuff-Rose Music, Inc.*,[91] copyright cases should be decided on a case by case basis because the Copyright Act does not favor the victim or the reuser.

At a minimum, however, a solution based on the economic model should be broad and general, yet guided enough for courts and software authors to use in practice. Concrete guidelines for lower courts, rather than abstract economic principles, are important for at least four reasons. First, summary judgment is extremely important in copyright infringement cases.[92] Second, reuse is quite often admitted or undisputed, in which cases the judge might likely determine what is reusable as a matter of law and what standards a jury should consider when determining illicit copying. Third, cases are often heard in a bench trial when statutory damages are at issue. Fourth, programs as a whole may be strikingly or substantially similar in the eyes of a jury, yet more specific guidelines may be needed to determine if some elements of the first program may not be reused.

A set of factors, rather than single rule, is an appropriate compromise in computer software copyright cases. It is true that a set of factors is administratively more costly than bright line rules, but given that the supposed bright line rules promulgated by courts are uncertain and inefficient if rigidly applied in practice, a set of factors is better than the status quo and perhaps the most certain that can be achieved by copyright law. A set of factors incorporating the economic model should provide guidelines that will achieve efficient economic goals.

The factors proposed below should be specific enough to guide both lower courts and computer software authors. This is not the first article to propose factors, but this article builds on prior decision-making methodology and provides a finer resolution to apply to different circumstances. For example, Dam suggests that the economic analysis boils down to (a) whether allowing reuse will substantially advance the state

90. *See* Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960). *See also* William H. Wright, Note, *Litigation as a Mechanism for Inefficiency in Software Copyright Law*, 39 UCLA L. Rev. 397, 436 (1991) ("But copyright law is not well suited to per se rules. [Analysis] should be performed on the basis of the facts of each dispute.").

91. Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 577 (1994).

92. Clapes, *supra* note 16, at 963.

of technology, and (b) the effect of reuse on authors' incentives.[93] Others have made the analysis and suggest applying patent-like standards to copyright in order to encourage innovation.[94] Still others would look at design methodology or behavior.[95]

What the literature does not provide, however, is guidance in determining whether technology actually has been advanced, whether a program is innovative enough to meet a patent-like standard, or whether the program's design or behavior warrants protection in any given case. Dam's simplification, for example, works well in theory, but is difficult to apply in practice because some facts point toward technological progress while others point toward non-progress. Each possible test Dam presents is offset by considerations which show that the test may not be efficient.[96]

The court must weigh more finely granulated factors to determine how to actually dispose of a case. For example, in *Atari v. Nintendo*, the court had to consider (a) reverse engineering for competition, (b) a potential monopoly, (c) wrongfully obtained source code, and (d) a product that arguably caused technological progress, all before determining that the reuser was liable for copyright infringement.[97] In another difficult case, *Vault Corp v. Quaid Software, Ltd.*,[98] the court ruled that a program that defeated a software "lock" was not a copy because it performed the opposite "unlocking" function. This result would understandably affect incentives to create both locking software and software generally, but might spur development of better locking programs. The goal of the factors is to make cases like these easier to decide.

B. Deriving the Factors

In each of the factors below, analysis of the model essentially leads to the same conclusion: societal benefits are either increased or decreased, and the costs and benefits to the parties are roughly equal. While this truism might make the factors appear worthless, the opposite is true. Because societal benefits go up or down under each and every

93. Dam, *supra* note 16, at 363-64.

94. Mark A. Lemley, *The Economics of Improvement in Intellectual Property Law*, 75 Tex. L. Rev. 989, 1083 (1997); Menell, *supra* note 72, at 1095.

95. *See* Dennis M. Carleton, Note, *A Behavior-Based Model for Determining Software Copyright Infringement*, 10 High Tech. L.J. 405 (1995) (arguing for broad protection of computer software based on "behavior" of each computer program); Beutel, *supra* note 7 (stating that copyright scope should be defined by the software design phase, giving more protection to elements that are added later in the design process).

96. *See, e.g.*, Dam, *supra* note 16, at 364 ("Even if this approach might lead one to permit copying. . . two important considerations point in the opposite direction."). Dam notes that case by case determination may lead to uncertainty in property rights.

97. Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 834 (Fed. Cir. 1992).

98. Vault Corp. v. Quaid Software, Ltd., 847 F2d 255, 268 (5th Cir. 1988).

case the court hears, it is that much more important to be able to distinguish cases (or fact patterns within a case) where societal benefits increase and where societal benefits decrease. The factor-based analysis becomes even more important when multiple factors appear in the same case, and the court must weigh each factor against the others.

Each of the factors is essentially a categorization and generalization of certain facts which are not knowable before creation of the first work, but are important and known at the time reuse occurs. A key question is why the generalizations do not simply lead to new *ex ante* rules that affect incentives to create. The answer is that they do and they do not. To some extent, knowing each of the factors will help a potential software author decide both how much to create and how much to reuse. The reuse allowed by the factors balances the costs and benefits of copyright protection at the time the first work is created. At the same time, however, because the specific state of affairs is unknown until the reuser actually makes the decision to reuse rather than at the time the first program is created, full determination under the factors remains unknown until the time just prior to reuse. Because there are two actors, the court must apply the factors in such a way that *ex ante* incentives to create and *ex post* incentives to reuse maximize social benefits.

To the extent that one party is unable to discern the exact behavior of others or cannot cheaply guess what will happen on average, imposing liability based on what actually happens may maximize social benefits. Generalized factors are a categorical assessment of certain costs and benefits that can have an effect similar to *ex ante* rules; at the same time, however, *ex post* determination of those factors may maximize social benefits where there are two actors.

An analogy in tort law is per se negligence. While courts could determine in each specific instance whether the cost of certain behavior outweighs the costs of an accident, they instead categorize certain violations of law where negligence will be assumed. This "factor" affects incentives to act carefully because the person acting in violation of the law can plan his or her affairs; he or she knows what the likely outcome at trial would be. At the same time, the tort victim may not know that the defendant is breaking the law. While the factor affects the *ex ante* behavior of one party, it only affects the other party's behavior based on expected behavior or based on game theory guesses about what the other party will do. Thus, the *ex ante* part of the factors will affect the defendant, and an *ex post* analysis might be necessary to influence the plaintiff. This result is consistent with the game theory description of the economic model and the predictions about incentive effects associated with the model.

C. The Factors

There are four basic factors that categorize the model and should guide the courts.

1. *Market Substitution*

An important factor for courts and authors to consider is whether the reuser has substituted the new work into the market in place of the pre-existing work. If the reuser experiences a dollar gain and the earlier author experiences a dollar loss, then the incentive to create would be eviscerated.[99] If the ratio of substitution is less than 1:1, then a court would accordingly be more likely to allow reuse, depending in part on other factors.[100]

This factor is already explicitly considered by courts, and is therefore not particularly new to the analysis. In *Campbell v. Acuff-Rose Music, Inc.*, the Court discusses transformative versus substitutive reuse in the context of parody.[101] Under this factor, reuse that transforms but does not substitute for the first work is not considered infringing. Dam notes that in order for this factor to be meaningful, the enhanced features must be significantly important, and the value added by the enhancements must be a substantial portion of the value of the second product.[102] This factor is represented in the model as:

a. ΔB_p is negative due to profits taken by the reuser.
b. ΔB_r is positive and equal to or greater than the decrease in ΔB_p depending on how much value the new product adds in the market.
c. ΔB_s is ambiguous depending on the amount of substitution. One component is negative due to decreased incentive to create caused by decreased profits. An offsetting component is positive due to increased competition and the innovation that follows new creation. The more the substitution, the more incentives decrease and the smaller the benefit from competition.

On the whole, to the extent that societal and producer benefits decrease more than benefits increase for the reuser, the court will disallow reuse.

a. Originality v. Slavish Copying

One subset of market substitution is "slavish copying." Dam and

99. Piracy is the penultimate 1:1 substitution.

100. *See* David W.T. Daniels, *Learned Hand Never Played Nintendo: A Better Way to Think About the Non-Literal, Non-Visual Software Copyright Cases*, 61 U. Chi. L. Rev. 613, 637 (1994) (stating that programs that reduce incentive to create should be found infringing).

101. Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 591-93 (1994).

102. Dam, *supra* note 16, at 362.

others call this "me too" copying.[103] It stands to reason that if the economic goal of copyright law is the creation of original works, then the more original/innovative the product associated with a certain reuse is, the more likely a court will allow it.[104] Conversely, if the reuse is a slavish copy, then reuse will likely be disallowed.[105]

In a survey of 25 cases between 1978 and 1990, Soma, et. al., found that 15 cases involved total copying of a substantial portion of the first program, and that of those 15 all but one were found infringing.[106] The saving grace for the lone winning reuser was a failure to place a copyright notice on a ROM chip; this decision would be different after the Berne Convention.

Consistent with *Feist*, courts require more than just "sweat of the brow" before they will allow reuse of software elements. If an entire user interface is copied, regardless of its simplicity,[107] extensive original source code programming that makes the software work will not save the reuser from infringement liability. On the other hand, if some originality is shown that creates technological advancement and modifies the first work in some way, then reuse will more likely be allowed.[108] The more the original work is modified, the more likely reuse will be allowed.[109]

Under the model, this is essentially the same as 1:1 market substitution, except there is also no offsetting social benefit gain from increased competition because slavish copying will not create competitive pressure to increase the supply of goods or improve products.

2. *Breach of Economic Relationship*

If the reuser breaches an economic relationship with the first author, a court is more likely to disallow reuse or alternatively to assess damages.[110] Soma found that in 10 cases where a party breached an

---

103. *Id.* at 358.

104. Landes and Posner, *supra* note 2, at 344 (noting that the source of literal copies will garner more protection than the source of derivative works).

105. *See e.g.*, Fonar Corp. v. Domenick, 105 F.3d 99 (2d Cir. 1997).

106. Soma, et. al. ., *A Proposed Legal Advisor's Roadmap for Software Developers: On the Shoulders of Giants May No Breachers of Economic Relationships Nor Slavish Copiers Stand*, 68 DENV. U. L. REV. 191, 220-23 (1991).

107. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 78 (D. Mass. 1990).

108. *See* Glynn S. Lunney, Jr., *Lotus v. Borland: Copyright and Computer Programs*, 70 TUL. L. REV. 2397, 2403 (1996) (arguing that copyright should protect programs only where there is an "undue" advantage to competitor).

109. *See* NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d (BNA) 1177, 1186-87 (N.D. Cal. 1989), (allowing reuse even though microcode programmer assumed to have started with copied code modified it until the new work and the first work were no longer similar.)

110. *See, e.g.*, Computer Assocs. Int'l, Inc. v. Altai, Inc., 592 F.2d 718 (breaching of confidentiality obligation gives rise to trade secret claim even if no copyright infringement);

economic duty, courts found liability in all 10 cases, regardless of the degree of copying and even in cases where the reuser did substantial original authorship to create the second product.[111]

The *ex post* economic basis for this factor is that a court does not want to allow a "breacher" to gain without having to pay for it.[112] As noted in the discussion about *Gracen*, an *ex ante* analysis indicates that a low level of protection is all that is necessary, because the first software author would not provide the fruits of his or her labor to the reuser without pay, and that pay would cover the risk of reuse.[113] For example, in *Effects Associates, Inc. v. Cohen,* the court held that failure to pay for movie special effects footage did not raise copyright issues because of an implied license to use the footage.[114]

This analysis, of course, depends in large part on the *ex ante* assumptions of the parties. With software, parties are unable to make assumptions because the degree of reuse allowed by courts seems to differ based on whether an economic relationship exists which was breached after entering the relationship and the parties will not know before entering the relationship which state of the world they will be in. Thus, they may not have an accurate estimate beforehand and may not be able to protect themselves by contract. Further, as discussed below, it is not always possible to set a price on the disclosure of trade secrets, and thus the parties might not be able to contract in advance for a breach.[115]

A court reviewing the matter *ex post*, however, sees that a relationship has been breached and has two options to create incentives for parties to contract and breach efficiently in the future. First, the court can assess some sort of contract/quasi-contract liability but continue to allow reuse. Second, the court can disallow reuse that might otherwise have been allowed. These choices are not equally efficient due to Arrow's information paradox: in the case of source code and other trade secret elements, the act of reuse may not be compensated by contract type damages because the other party would be free to exploit the technology in ways uncontemplated by the original agreement, and the very act of

Lasercomb Am., Inc. v. Reynolds, 911 F.2d 970, 971-72, 980 (4th Cir. 1990) (barring infringement suit due to copyright misuse but allowing damages for fraud based on representations that licensee would protect licensor's copyright).

111. Soma, *supra* note 11, at 220-23. Soma did not tie the breach to economic incentives to create and contract. *See also* Kepner-Tregoe v. Leadership Software, 12 F.3d 527, 531-32 (5th Cir. 1994) (infringement was found where former partner created similar work to compete contrary to license agreement).

112. POSNER, *supra* note 2, at §4.8.

113. Source code, for example, is usually kept as a trade secret as well as a copyrighted work.

114. Effects Assocs., Inc. v. Cohen, 908 F.2d 555, 558-59 (9th Cir. 1990).

115. For example, many arbitration clauses exempt injunctive relief from arbitration.

contracting creates this risk with no way to protect against reuse.[116] This distinguishes *Effects Associates* from software cases.

While only contract remedies were available in *Effects Associates*, if the work was software—especially source code—the outcome might have been different. In *Cadence Design Systems, Inc. v. Avant! Corp.*, the court held that a likelihood of infringement entitled the owner of the pre-existing work to a preliminary injunction, even if the reuser might go out of business, and even if money damages were adequate.[117] This is because social cost of decreased incentives to create and license copyrighted programs are high enough such that simple expectation damages will not induce social wealth maximizing behavior; additional damages are required to create the incentive. With copyright remedies available, the court can choose a higher level of copyright protection, and can set damages or enjoin reuse as necessary to obtain an efficient outcome. Under the model, this is represented as follows:

a. ΔC_p is positive due to the economic breach, and could be quite large if source code is involved.

b. ΔB_r is positive and probably larger than ΔC_p. If ΔB_r were less than ΔC_p, then ordinary contract remedies equal to ΔC_p, if available, might be sufficient to deter breach.

c. ΔB_s is negative due reduced incentive to contract or create where obligations may be breached.

Thus, when societal benefits are reduced and production costs increased more than the reuser can gain, reuse will not be allowed.

3. *Customer Need for Compatibility*

Under this factor, the more costly it would be for customers were reuse disallowed, and the more benefits customers could obtain were reuse allowed, the more likely a court will allow reuse. This factor is accentuated by the *ex post* analysis. It is not until after customers have used a product that a court can assess the actual costs and benefits involved. While an *ex ante* analysis would look at the expected customer needs and make predictions and rules based on such guesses, courts look at the *ex post* outcome to determine what customers have actually done.

As long as slavish copying of an entire program is disallowed, this factor still ensures that companies recoup their investment in the first software program. They will, of course, make investment decisions which encompass the probability that their program will lose protection of some program elements due to customer needs. Thus, the investment in each program may decrease, but the ability for new program authors

116. Lemley, *supra* note 94, at 1050-51.

117. Cadence Design Sys., Inc. v. Avant! Corp., 125 F.3d 824, 827-29 (9th Cir. 1997), *cert. denied* 118 S. Ct. 1795 (1998).

to compete and add new features by reusing certain elements offsets this decrease. Because protection is still extended to the work as a whole, this result is different than the "truncated protection" proposals discussed above.[118] Even so, a court will only reduce the protection if customer—that is, societal—interests would actually be served by reducing protection.

This factor bears on two controversial ideas from the literature: "switching costs" and "de facto" standardization.[119]

a. Switching Costs

Switching costs are the costs a user incurs by switching from one program to another.[120] As noted above, loss of the use of data files is one cost. In addition, costs include purchase (or license fee) of the software itself, retraining, and new hardware necessary to run the software. Dam notes that the analysis of switching costs differs where software only is involved, as opposed to the purchase of new computer hardware in addition to computer software.[121] Thus, in order to switch to a new software program, a user had to consider both hardware and software costs, and high hardware costs might force a user to buy inferior and more costly software in order to avoid purchasing hardware. To the extent, however, that computer hardware costs have dramatically decreased in relation to software costs and to the extent that faster computer hardware is necessary to run newer software programs at all, this distinction is not necessary.

As switching costs increase a court would be more likely to reduce protection. This sliding scale allows more flexibility than Clapes, who argues for broad protection, reasoning that switching costs are overestimated generally,[122] or Larvick, who argues that consumer benefits of standard interfaces are high enough such that no copyright protection should apply to user interfaces.[123] A factor based analysis allows the court to determine whether or not switching costs are actually high enough to reduce protection.

118. *But see* Brown, *supra* note 22, at 835 (holding that "market factors" should not be considered in copyright protection). The argument here, however, is not that protection ceases because a product is successful. The argument is that market success causes customer reliance, and that limited reuse of key program elements may be efficient.

119. Dam, *supra* note 16, at 346.

120. *Id.*

121. *Id.* at 349. For example, until recently only Apple made Macintosh computers.

122. Clapes, *supra* note 16, at 961.

123. Larvick, *supra* note 47, at 211, 215.

b. De Facto Standards

A product becomes a "de facto" standard when virtually everyone uses it even though no governing body has approved the product as a standard.[124] Some have proposed that protection for a program cease as soon as it becomes a de facto standard. This argument suffers from the same weakness as proposals to cease protection once investment is recovered: software authors don't know beforehand if their product will become a standard.[125] Patry points out that this rule would also be empirically problematic due to generally short lived market leads; he points out that by the time of the *Lotus v. Borland* decision, Lotus was no longer the substantial market leader it had once been.[126]

The broad proposals to cease protection refer to wholesale copying of de facto standard programs which would be inefficient and thus disallowed under the slavish copying factor here. Reuse of certain elements of de facto standard programs, however, presents a different issue than wholesale copying, and does not suffer from the same inefficiency.

Dam asserts that the *scenes a fair* doctrine[127] will serve to allow use of popular program elements, so focus on de facto standards is unnecessary.[128] Dam's argument, however, begs the question: how is a court supposed to know when a program element is so widely used that it should no longer be protected? Because Dam focuses primarily on wholesale copying he need not confront this issue, but reuse today includes both reuse of elements as well as complete appropriation. Someone had to use each element first, and arguably that person would not want copyright protection in his or her element to cease.

The question is whether there exists a trademark-like "policing" duty that the first creator must perform to keep a program element from becoming an industry standard, or whether a program element is so important that courts will allow reuse whether or not the element has been "policed." Both types of cases have been before courts, and reliance on the *scenes a fair* doctrine without guidance about when the doctrine should actually be applied has little practical use. The "customer compatibility needs" factor, however, allows courts to assess—at any time

124. MS-DOS on PC computers was, and still is in many ways, the de facto standard operating system on PC computers.

125. Dam, *supra* note 16, at 351.

126. Patry, *supra* note 39, at 8 n.37.

127. In the literary field, *scenes a fair* refers to stock literary devices that are part of every author's repertoire. As applied to computer software, the doctrine limits protection for elements that are either necessary to achieve a certain end (such as a grid in an electronic spreadsheet), are standard for the industry (such as a menu bar), or follow from a common theme (such as a desktop). The *Gates* court provides a good definition. Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 838 (10th Cir. 1993).

128. Dam, *supra* note 16, at 360.

during the first product's life cycle—when an element should cease to be protected and when it may be used by other programmers. At the point in time when the product becomes a de facto standard, the creator would no longer be able to rely on the product's "momentum" and others would be able to use certain elements.[129] It may be, like many generic trademarks, that early on in a product's life the element is protected but later on the element may be reused. Under the model, this factor is represented as follows:

a. ΔB_p is negative due to market substitution for the new product.

b. Assuming no slavish copying, ΔB_r is positive and greatly outweighs the negative ΔB_p due to the reuser's ability to break into a market that would otherwise be captured by the producer's product. Unlike the inefficient proposal to eliminate all protection when a producer recovers his or her costs, limited copying of specific elements necessary to reduce switching costs will not have an incentive killing impact on the producer's benefits, B_p. The producer has likely reaped monopoly-like benefits if its program is so widely used as to create a need for compatibility.[130]

c. Part of ΔB_s is negative due to a lack of incentive to create. The decrease is not large, however, because ΔB_p is small.

d. Part of ΔB_s is positive due to increased competition and incentive to innovate and more customers who are better able to choose a program that meets their needs.[131]

e. One component of ΔC_s is low due to decreased switching costs.

f. Another component of ΔC_s may be high or low, depending on the quality of the pre-existing program. If the program is of high quality, then the increase in social costs will be low by allowing reuse; barring reuse may actually create social costs.[132] If the program quality is low, then the increase in social costs will be high because allowing reuse will not create incentives to create improved technology.[133] While there is disagreement on this issue,[134] the partial reuse for customer compati-

129. Timothy S. Teter, Note, *Merger and Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases*, 45 STAN. L. REV. 1061, 1072 (1993); Joseph Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35, 36 (1989).

130. Lawrence D. Graham and Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection and Disclosure*, 22 RUTGERS COMPUTER & TECH. L.J. 61, 125 (1996) (noting broad approval of reverse engineering when monopoly profits are large).

131. *Id.*

132. Nicolas P. Terry, *GUI Wars: The Windows Litigation and the Continuing Decline of Look and Feel*, 47 ARK. L. REV. 93, 132 (1994).

133. Farrell, *supra* note 129, at 46 (stating that excessive dissemination means that an inferior standard may capture the market).

134. *Compare* Farell, *supra* note 129, *with* Larvick, *supra* note 47, at 211-12 (limited or no protection for user interfaces will not "freeze" current interfaces into static standards), *and* Menell, *supra* note 72, at 1068 (holding that broad copyright protection may lead companies to adopt incompatible and non-efficient standards to avoid reuse).

bility indicates that the increase in social costs would be small, because new software manufacturers will be allowed only to reuse the elements necessary to keep customer costs low, but reusers would otherwise add functionality in order to obtain future customers.[135]

g. The gain in social benefit and the decrease in social cost should outweigh the sometimes administratively costly determination of whether customer needs for compatibility justify limited copyright protection for certain elements.[136]

On the whole, social benefits are increased, social costs are increased slightly, and benefits gained by the reuser outweigh a loss in benefits to the producer.

4. *Competitive Need for Compatibility*

This factor, often called "functional compatibility," measures the necessity for reuse based on the functional requirements of the computer and the desired application. The factor is similar to the trade dress notion of functionality: to the extent that software looks similar because of functional requirements, the reuser will not be held liable for copyright infringement.[137] This factor is an offshoot of the idea/expression merger doctrine expressed in *Herbert Rosenthal Jewelry Corp v. Kalpakian*.[138] For computer software, this factor is explicitly discussed *in NEC v. Intel*, where the court states "[h]aving conceded at trial that NEC had a right to duplicate the hardware . . . Intel is in no position to challenge NEC's right to use the aspects of Intel's microcode that are mandated by such hardware."[139]

This factor is usually most important where the program element is unseen by competitors or unique to a product's function, and the only way for other companies to compete is to reuse at least part of the first program. Reuse may be the only way some companies can survive in a competitive market.[140] This is a factual question that must be deter-

135. *See, e.g.*, Menell, *supra* note 72, at 1069 n.131 (noting that a file conversion program—as noted above a form of reuse generally allowed in market equilibrium—may alleviate the problem of standard adoption in general).

136. Teter, *supra* note 129, at 1072.

137. In Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 148-49 (1989), the Court held that the patent system alone governs copying of ideas and functional elements.

138. Hebert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742-43 (9th Cir. 1971) The court limited protection of a pre-existing jeweled bee broach because of the limited number of ways that jewels can be put together to look like a bee.

139. NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d 1177, 1188 (N.D. Cal. 1989).

140. Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091, 1093 (1995); Menell, *supra* note 72, at 1067-68.

mined *ex post*.[141]

Some argue that competitive compatibility is simply a pretext for cheaply taking profits rightly earned by the first author's original creation.[142] This argument fails economic analysis, however.[143] Companies that must copy in order to compete under this factor will usually attempt to license the technology because it is less expensive and more expedient than reverse engineering and protracted legal battles.[144] Because one company owns the technology another is trying to license, however, bilateral monopoly costs arise which make voluntary and efficient transactions more difficult to achieve. Courts would step in to correct the market failure; only when efforts to license at a cost efficient price fail or are unattainable may reuse of essential elements occur. Under the model, this factor is represented as follows:

a. ΔB_s is positive due to increased competition and increased value to computer program users.[145]

b. As with the other factors, ΔB_r is positive and ΔB_p is negative, and they roughly offset each other, though the benefit gains to the reuser will likely exceed the benefit loss to the producer based on increased ability to compete. Of course, the more elements of a program that are reused, the more equal redistribution of elements becomes.

c. One part of ΔB_s is negative due to decreased incentives to create. This decrease in social benefits is outweighed by the gained benefits of increased competition so B_s will increase.

d. Like the customer needs factor, it is also possible that C_s will increase greatly due to widespread dissemination of an inferior standard. On the other hand, if the court examines which elements are truly necessary to compete and allows reuse of only those elements, then it is likely that technology will advance due to new elements added by the reuser. Because the primary benefit of this factor is market entry, it is reasonable to expect that new competitors that do not slavishly copy will add new program elements in order to actually make a profit.[146]

---

141. *See* Engineering Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1347 (5th Cir. 1994) (remanding case to district court to hear evidence on the extent that program input and output formats by market requirement).

142. Christopher Hager, Note, *Apples & Oranges: Reverse Engineering as a Fair Use After Atari v. Nintendo and Sega v. Accolade*, 20 RUTGERS COMPUTER & TECH. L.J. 259, 320 (1994) (". . . their true objectives were to cash in on commercial markets established by the plaintiffs . . . .").

143. David A. Rice, *Sega and Beyond: A Beacon for Fair Use Analysis . . . At Least as Far as It Goes*, 19 U. DAYTON L. REV. 1131, 1146 (1994) ("The Ninth Circuit concluded, however, that the public benefit resulting from a commercial use is a factor worthy of consideration even if the objective and consequences of the use is economic gain.").

144. Lauren Bruzzone, Note, *Copyright and License Protection for Computer Programs: A Market Oriented Assessment*, 11 PACE L. REV. 303, 314 (1991).

145. Teter, *supra* note 129, at 1063-71.

146. For example, dropping prices of software developer kits (many are now free) programmers use to develop "add-ons" to existing software products indicates that strong

D. Justifying the Factors: Court Decisions

A second and more important test of the model is whether courts are actually making decisions based on it. One way to verify whether the model is useful and not simply an after the fact rationalization is to see if it reconciles seemingly irreconcilable cases. The following discussion and comparison of cases supports the application of the decision factors to actual cases.

1. Lotus v. Borland *and* Whelan v. Jaslow

Perhaps the most difficult and most important set of cases to reconcile is *Lotus v. Borland*, which grants almost no protection, and *Whelan v. Jaslow*, which grants seemingly endless protection. Using the economic decision factors, the result is relatively clear. In *Whelan*, the reuser had entered into an economic relationship with the owner of the first program and breached that relationship by using arguably un-copyrightable structure, sequence, and organization of the program to make a new program. The court saw this breach and ruled, implicitly at least, that as between the two parties to an agreement, the non-breacher should win.[147] Breaching an economic relationship appears to be the singularly dispositive factor, as none of the other factors were present in this case.[148]

In *Lotus*, on the other hand, no such relationship existed: the parties were competitors. In this case, Borland used a "command interpreter" that allowed Lotus 1-2-3 users to run macro scripts and to enter Lotus 1-2-3 command keystrokes.[149] The importance of these keystrokes to users was enormous. Not only had users built up training in operating the program,[150] but macro scripts are also written as command short-

---

copyright protection is not necessary to advance new standards. By cheaply and freely licensing computer code that might otherwise be protected, the companies are encouraging externalities that will make one product the "standard." Because developers do not need to pay, they will support, without reference to copyright protection, what they view to be the "better" or at least customer preferred standard.

147. Patry, *supra* note 39, at 4. (*Whelan* was a "compelling case of infringement" based on unauthorized reproduction by a former business partner.).

148. Soma, et. al., *supra* note 106, at 220-23 supports this conclusion. *See also* Manufacturer's Technologies v. CAMS, 706 F. Supp. 984, 993-94 (D. Conn. 1989), where the court rejected arguments that competitive need justified reuse where the defendants developed plans for a new program while employed by the plaintiff as sales representatives.

149. Lotus uses the "slash" key to invoke command menus, and then uses the first letter of the command as a short-cut. For example, a user typing "/fo" would invoke the [F]ile [O]pen command.

150. WordPerfect, for example, changed its keystrokes when it moved its word processor from DOS to Windows. "Shift F7" is the print command in DOS and "Control P" is the print command in Windows. In order to maintain user training in its DOS program, however, WordPerfect provided with its Windows version an option to use the same keystrokes avail-

cuts.[151] Thus, thousands of users would have lost uncountable training time and productivity enhancing macro scripts by switching to Borland's program if Borland was unable to reuse the portion of the Lotus 1-2-3 user interface. As a result, the "customer needs" factor was considered by the court. Because of the near monopoly that Lotus held,[152] the switching costs for customers for both macro compatibility and training costs, in addition to enhanced competition, greatly outweighed any cost to Lotus.

Lack of the slavish copying factor mitigated the reuse by Borland. Borland did not reuse entire portions of the Lotus 1-2-3 interface and it completed extensive work in both creating its own interface and in adding features not offered by Lotus. Compare this result with *Lotus v. Paperback*, where the defendant had not only reused elements necessary to meet customer needs but had also slavishly copied the entire Lotus 1-2-3 interface with few improvements to the software.[153] There the customer needs factor was outweighed by slavish copying.

Note that the "competitive needs" factor was not considered in *Lotus v. Borland*. Borland could have, and until the First Circuit reversed the district court had, released a competing spreadsheet without any reuse of the Lotus program.

2. *Substitutive Use as a Dispositive Factor*

As the two *Lotus* cases above indicate, substitutive use—and even slavish copying—is not dispositive of reuse that would be considered copyright infringement. When other factors are not present, however, slavish copying will virtually always be considered infringement.

In *Digital Communications Assocs., Inc. v. Softklone Distrib. Corp.*,[154] the district court disallowed reuse of a text based screen which looked almost exactly like a copyrighted work even though the screen

able in the DOS version; the author notes that six years after abandoning the DOS version of WordPerfect, he still uses the DOS keystrokes.

151. Thus, a macro to open a file would be written "/fo" just as the command to open a file would be entered by a user.

152. Brian Johnson, Comment, *An Analysis of the Copyrightability of the "Look and Feel" of a Computer Program: Lotus v. Paperback Software*, 52 OHIO ST. L.J. 947, 948 (1991) (estimating a market share of 70% - 80%).

153. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 78 (D. Mass. 1990)

> Thus, defendants had no choice, they argue, but to copy these expressive elements from 1-2-3. Had they not copied these elements (including the macro facility), users, who had been trained in 1-2-3 and had written elaborate macros to run on 1-2-3 spreadsheets, would be unwilling to switch to VP-Planner. VP-Planner would be a commercial failure. Neither the factual nor the legal predicate of the argument is supportable.

Id.

154. Digital Communications Assocs., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 460 (N.D. Ga. 1987).

was very simple and arguably not original. While the court could have easily held that the work was not copyrightable, it instead provided protection to the work as a whole and found infringement due to slavish copying.

3. *The Video Game Cases*

Two cases relating to video game consoles illustrate how the factors are weighed against each other. In *Sega Enterprises, Ltd. v. Accolade, Inc.*,[155] the Ninth Circuit held that substitutive use in reverse engineering was acceptable where intermediate copying of an entire program was necessary in order to create a competing video game program cartridge compatible with a single game console. In this case, the competitive need for compatibility outweighed the substitutive use.[156]

As in *Sega*, in *Atari Games Corp. v. Nintendo of America, Inc.*[157] Atari was subject to onerous license terms from Nintendo prior to reverse engineering Nintendo's code.[158] It would appear at first that this case was the same as *Sega*; the court relied in part on *Feist*[159] and reasoned that reverse engineering was generally acceptable.[160] However, the court found that Atari lied to the Copyright Office to obtain a copy of Nintendo's code while it was a licensee of Nintendo and also found that Atari copied more than was necessary to compete.[161]

Thus, breach of the economic relationship and substitutive use factors outweighed the competitive need for compatibility.[162] In fact, as noted above, the breach of an economic relationship was probably dispos-

---

155. Sega Enterprises, Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

156. *Id.* at 1514 ("Prior to rendering its own games compatible with the Genesis console, Accolade explored the possibility of entering into a licensing agreement with Sega, but abandoned the effort because the agreement would have required that Sega be the exclusive manufacturer of all games produced by Accolade.").

157. Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832 (Fed. Cir. 1992).

158. *Id.* at 836

> The license terms, however, strictly controlled Atari's access to Nintendo's technology, including the 10NES program. Under the license, Nintendo would take Atari's games, place them in cartridges containing the 10NES program, and resell them to Atari. Atari could then market the games to NES owners. Nintendo limited all licensees, including Atari, to five new NES games per year. The Nintendo license also prohibited Atari from licencing NES games to other home video game systems for two years from Atari's first sale of the game.

Id.

159. Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 349 (holding that pure fact is not copyrightable).

160. *Atari Games Corp.*, 975 F.2d at 844.

161. *Id.* at 841, 845.

162. *See* Mark L. Gordon, *Copying to Compete: The Tension between Copyright Protection and Antitrust Policy in Recent Non-Literal Computer Program Copyright Infringement Cases*, 15 JOHN MARSHALL J. COMPUTER & INFO. L. 171, 176 (1996).

itive of an infringement finding in this case.[163] As noted by Gordon, the key differences between *Sega* an *Atari v. Nintendo* were that the *Sega* code was purely functional and that in *Sega*, the reuser had no other way to create a compatible program and did not obtain an unauthorized copy of the source code.[164]

4. *The Speed-Up Chips*

In *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc*,[165] the reuser (Galoob) made a game cartridge add-on that intercepted certain pieces of data between Nintendo game consoles and cartridges, and either edited or blocked the data to change game play, such as allowing unlimited lifetime of game characters. The court held "it does not necessarily follow that kaleidoscopes create unlawful derivative works when pointed at protected artwork. The same can be said of countless other products that enhance, but do not replace, copyrighted works."[166]

In *Galoob* the court relied on a hybrid of the customer need for compatibility factor and the competitive need to compete factor. Like a kaleidoscope, Galoob's "Game Genie" was used to fulfill a certain, though perhaps unusual, market demand. Whereas kaleidoscopes fill a market need for distorted vision, Game Genie fulfilled a need for unlimited video game lives. Switching costs would have been high for customers in each case. Kaleidoscope users would have to severely limit what they look at, and game players would have to purchase a new game system from a manufacturer willing to license the derivative work right to Galoob. The factor intertwines with the competitive need as well, because the only way for the kaleidoscope maker and Galoob to fulfill the market niche is to reuse some amount of the first work.

The application of these factors in *Galoob* indicates that the social benefits of the development of new products and technologies, no matter how unusual the market may seem for those technologies, outweighs the social and private costs of reuse where no slavish copying occurs, especially when there is little or no substitution. In *Galoob*, the court rejected the argument that the product at issue usurped Nintendo's ability to create a similar product, and held that the Game Genie did not reduce the market for any existing game.[167]

Compare *Galoob* with *Midway Manufacturing Co. v. Arctic International, Inc.*, where the court held that a "speed up" chip allowing com-

163. *Compare with* Harper & Row Publishers, Inc. v. Nation Enterprises, 471 U.S. 539, 541, 553 (1985) (finding infringement where printing of excerpts caused exclusive licensee to terminate contract with copyright owner).

164. Gordon, *supra* note 162.

165. Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc., 964 F.2d 965 (9th Cir. 1992).

166. *Id.* at 969.

167. *Id.* at 971.

puter game characters to move more quickly was a derivative work, and thus infringing.[168] In *Midway*, the chips were both replacements for the first chip and a potential future market substitute. The court noted that game manufacturers would want to exploit the new "speed up" market and that the infringing chip destroyed this opportunity.[169]

In *Midway*, the court had no information on the new market because such a market had not yet developed and so the court made a determination that market substitution would harm prior authors and reduce incentives to enter that market. In *Galoob*, the court had *ex post* evidence that Nintendo had no intention of exploiting the known new market, and consequently considered the new facts, downplaying the market substitution factor in favor of the competitive need to reuse in order to exploit a market that the copyright owner was not exploiting.

5. *Economic Obligations v. Future Work for Authors*

In *MAI Systems v. Peak Computer*[170] the Ninth Circuit held that a third party who loaded a computer program into a computer system's internal memory made an impermissible copy. To many, this seemed like an unsupportable result.[171] However, the factors make clear why the court made its decision. MAI had service agreements with its customers that required all service to be done by MAI technicians.[172] When the customers allowed Peak to service the computers, they breached an economic obligation to MAI, and thus the court disallowed the reuse by Peak.

In *Brown Bag Software v. Symantec, Corp.*,[173] a programmer named John Friend created an outlining program for Brown Bag called PC-Outline. After leaving Brown Bag, Friend created another outlining program for Symantec, called Grandview. A review of the two programs showed that many of the elements were similar, but the court found no infringement. The court did not apply the breach of economic obligation factor because Friend was no longer under any obligation to Brown Bag. If anything, it applied an opposite factor not discussed above: authors must be able to work again in their area of expertise, or they will not create in the first place.

Further, the court relied heavily on the functional need for compatibility by upholding the district court's use of the idea/expression and

168. Midway Mfg. Co. v. Arctic Int'l, Inc., 704 F.2d 1009, 1013 (7th Cir. 1983).
169. *Id.*
170. MAI Sys. Corp. v. Peak Computer Corp., 991 F.2d 511, 519 (9th Cir. 1992).
171. *See, e.g.*, Bradley J. Nicholson, *The Ghost in the Machine:* MAI Systems Corp. v. Peak Computer, Inc. and the Problem of Copying in RAM, 10 HIGH TECH. L.J. 147, 167 (1995).
172. MAI Sys. Corp., 991 F.2d at 517.
173. Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1468 (9th Cir. 1992).

scenes a fair doctrinal tools, noting that most of the similar elements were those that were necessary to an outlining program.[174]

6. *Similar Subject Matter*

In *Data East v. Epyx,* the court held that the second author's karate game program did not infringe the first game.[175] The court held that in order to make a karate game, certain karate features would be necessary in each product.[176]

On the other hand, in *Broderbund Software v. Unison World, Inc.*[177] the second author created a program that created banners and cards in a manner similar to Broderbund's "The Print Shop." The court focused on the number of ways that "Choose a Font" menu could have been reworded and rejected the functional need for reuse.[178] While discussion of this one feature seems disingenuous given that the simple word "Font" is used in most word processors today, the court found on the whole that an "eerie resemblance" existed between the two programs' screens, sequencing, layout, choices, and feedback to the user.[179]

The programs in each case were in large part market substitutes. The cases show the extent to which the court is willing to apply the competitive need to reuse factor given the facts in the cases before them. In *Data East*, the court noted that both programs copied real world karate moves and needed similar features to implement the games, and thus the products could be similar.[180] The competitive need to reuse functionality trumped even a large substitution effect. In *Broderbund*, however, the reuser was not required by any real world or other functional need to select a set of functions, name those functions, and present those functions to the user in essentially the same way as the first author's expression. The court held the need to compete did not necessitate the need to reuse almost all of the structure, sequence, and organization.[181]

The comparison of *Data East* and *Broderbund* sheds light on the First Circuit's opinion in *Lotus v. Borland*. While the court held that Borland could reuse the menu hierarchy for the purposes of macro functionality, it did not disapprove of the *Lotus v. Paperback* holding, or even the *Lotus v. Borland* district court's holding for that matter, that a direct

174. *Id.* at 1472 -73. The features included pull down menus and a main editing screen to enter data.

175. Data East USA, Inc. v. EPYX, Inc., 862 F.2d 204, 208-09 (9th Cir. 1988).

176. *Id.*

177. Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127 (N.D. Cal. 1986).

178. *Id.* at 1134.

179. *Id.* at 1137.

180. *Data East*, 862 F.2d at 209.

181. *Broderbund*, 648 F. Supp. at 1133-34.

presentation of the entire menu with no changes was not allowed under the functional need to compete factor.[182] In other words, while the hierarchy itself was important for functionality, namely in the macro conversion, the presentation of that hierarchy in the form of a user interface was not important for functionality.

VI. APPLICATION OF THE MODEL: FUTURE CASES

The factor based solution proposed here guides analysis of future cases where the courts have yet to issue decisions.

A. Virtual Reality

Virtual reality, in its extreme form, is a computer program that immerses the user in the program. At its fullest implementation, it includes goggles that display three dimensional sights, earphones that play three dimensional sounds, and clothing that both interprets movement and sends tactile sensations to the user.

Virtual reality creates new questions of copyrightability for two reasons. First, virtual reality was designed to be object based. That is, virtual reality elements are highly conducive to reuse in other virtual "worlds" and authors design these elements to be reused.[183] Second, as technology advances, it will be more and more difficult to distinguish virtual reality from actual reality.[184]

The decision factors aid judges in determining the proper amount of protection.

1. *Direct Reuse of Specific Elements*

With respect to direct and complete reuse of specific elements—that is, copying—if the makers of the object really do intend that they be used in multiple "worlds," then the model implies that the market will not fail and the users will either license or freely allow reuse without bringing suit. Breaches of economic obligations arising from licenses would likely determine any copyright infringement issue in a case.

To the extent, however, that court cases do arise, then the decision factors would be used. Also, because only some elements will be reused,

182. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 810, 815-16 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996) ("The Lotus menu command hierarchy is also different from the Lotus screen displays. . . .").

183. Jack Russo and Michael Risch, Virtual Reality - A Legal Overview, in Computer Software: Protection/Liability/Law/Forms §19.05[1] (L.J. Kutten ed. 1994). For example, a virtual automobile might be used in a number of virtual cities. There is no need or desire of city designers to redesign virtual automobiles each time they create a new virtual city.

184. The best example of this is a *Star Trek: The Next Generation* holodeck program, which is frequently depicted as being indistinguishable from reality.

the slavish copying factor might not always be dispositive. For example, to the extent that a customer has previously licensed an element as part of a virtual world, and then reuses that same element in a second purchased or newly created virtual world, then the customer need for compatibility factor—based primarily on switching costs—would necessitate a finding of non-infringement. This would be especially true if the reuser relied on use of the first author's programming tools to create an element that is then reused in another author's virtual world.

In addition, the second author's competitive need to reuse might necessitate a finding of non-infringement. For example, if a first author creates a "hand" element that is widely used to navigate a virtual world, then the second author may not be able to compete in the market unless it can ensure its potential customers that it uses the same "hand" interface. In such a case, the customer externalizes its switching costs onto the first author, but the gains from a competitive product entering the market make up for these costs.

2. *Apparent Reuse of Realistic Elements*

With respect to distinguishing virtual reality from actual reality, the factors imply that the more realistic an element is, the more likely a court would require striking similarity or admitted copying before finding infringement.[185] This is because similarities in realistic elements would functionally be required to compete. Authors forced to create new reality based elements, not substantially similar to existing elements, might be locked out of the market due to the costs of such original creation. This would be truer to the extent that "cameras" could capture three dimensional technology objects, and future authors avoiding similarity would require original design work to depict the same object.[186]

On the other hand, striking similarity or slavish copying would mean complete market substitution and courts would have to look at the same factors discussed above under direct reuse of virtual objects. In addition, if an element is not realistic, competitors would not be able to gain a free ride off the work of others. While it might still be costly to produce non-realistic work, it would not be a cost that another competitor imposed by first capturing all rights to a realistic object.[187]

185. Russo and Risch, *supra* note 180, at §19.03; *but see* Gracen v. Bradford Exch., 698 F.2d 300, 304-05 (limiting protection rather than considering similarities). One way or the other, reuse is allowed absent some other factor.

186. Burrow Giles Lithographic Co. v. Sarony, 111 U.S. 53, 61 (1884) (holding a photograph is protected as original work); Bleistein v. Donaldson Lithographic Co., 188 U.S. 239, 249 (1903) ("Others are free to copy the original. They are not free to copy the copy.").

187. *See* Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340, 349-50 (1991).

B. Internet Copyright Infringement

Like virtual reality, copyright works on the Internet are based in large part on the reuse of protected elements. While downloading complete programs for use on a computer clearly falls within the slavish copying factor, more interesting is the treatment of smaller elements incorporated into documents on the World Wide Web.

The World Wide Web is an Internet standard where each document, or web page, is composed of several elements, including text directly included in the page, links that send the user to another page, and text or graphics incorporated from other sites on the Internet. A web page author could, if he or she wanted, completely reuse someone else's page without the knowledge of either the first page's author or the second page's viewer.[188] Until recently, web pages were purely documents, but now they also include programs that are written in a variety of computer languages.

1. *The General Case*

The core treatment of this type of reuse is clear. First, where a page is copied verbatim, courts would likely find the slavish copying factor and disallow reuse. Although many web pages are not created for commercial use, like other original works their creation is considered a social benefit and any complete copying that would tend to seriously harm the incentive to create will not be allowed. Second, much reuse is handled by the market. For example, many authors are able to provide free information to users by selling advertising "space" on their web pages. A page displaying information will also include an advertisement, which is in reality an element incorporated from some remote company's computer server. This transaction does not lead to market failure because the reusing web author has permission—and in fact is paid—to reuse an original element belonging to someone else. However, when a page is "framed" in another page, courts will be more hesitant to find infringement because no real substitution is occuring.[189]

2. *Use of Information*

A more difficult question is the use of unprotected information in web pages. While reusing the telephone book in *Feist* involved keypunch processing and was thus costly enough to encourage an attempt to contract, using information from remote computers on the Internet is a sim-

188. Trademark protection for web pages is beyond the scope of this paper.

189. *See* Futuredontics, Inc. v. Applied Anagramics, Inc., 45 U.S.P.Q.2d 2005 (C.D. Cal. 1998), (*aff'd* 152 F.3d 925 (9th Cir. 1998) (denying preliminary injunction for copyright infringement where one web page loaded another web page in a "frame").

ple as including a link in a web page.[190] In addition, because the information is by definition machine readable, manipulations of information are easily managed.

The decision factors give insight into how courts should rule on varying types of reuse. For example, reuse without modification of substantially all of the information available on a site would lead courts to disallow the reuse under the market substitution factor.[191] The fact that many web sites apparently license very small news stories from Reuters for reprint on their sites indicates that either (i) the companies believe that relatively small quantities of information would likely be protected or (ii) that reuse of the conglomeration of news stories would certainly be protected.

Where a reuser manipulates the information and represents it in a new format, the question is still more difficult. For example, a "fantasy baseball" web page might reprint baseball statistics obtained from a different author's web page and also perform calculations on the data to present adjusted scores.[192] The difficulty with this problem is that information is not really a computer program, and thus copying to compete or to avoid switching costs does not really apply. Of course, the duplicative cost of regathering information is a primary reason that "pure fact" is not protected in the first place.

In addition, because the information is transformed, there is very little market substitution. The first authors would argue that fantasy game players would have to visit their sites, download the information and manipulate it themselves. Because instant access to the processed information would be gone, it is likely, however, that users would look in newspapers for the information and avoid the first authors' sites anyway.

C. *Web Browser Standards*

Another development in the World Wide Web is the use of program components that are activated by web pages. A simple program is a marquee that scrolls sports scores across the screen. More complicated programs allow for complex installation and maintenance of other software on the computer.

Like OOP components discussed above, these program components do not cause a market failure. The companies authoring the program

190. For example, the link: http://maps.yahoo.com/yahoo/yt.hm?FAM=yahoo&CMD=GEO&SEC=geo&AD2=401+Florence+Street&AD3=Palo+Alto%2C+CA+94301&recent=0 would show a map to Russo & Hale LLP.

191. CCC Info. Serv. Inc. v. Maclean Hunter Market Reports, Inc., 44 F.3d 61, 61 (2d Cir. 1994).

192. *See* Kregos v. Associated Press, 937 F.2d 700, 708-09 (2d Cir. 1991) (holding that baseball statistics page is copyrightable, but receives limited protection).

components to be reused obtain enough benefit from the promulgation of the components that they allow others to reuse them inexpensively or freely. For example, Sun recently sued Microsoft, not alleging that Microsoft had copied its Java language components, but instead alleging that Microsoft's implementation of the Java components were not "accurate" and were harming Sun's reputation in the market.[193] If Sun saw value in stopping Microsoft's reuse altogether, it might be able to convince a court to ban reuse based on the breach of an economic obligation factor.

Competing World Wide Web browser software is a useful example of many of the factors. First, it illustrates that copyright protection in general does not lead to proliferation of a single standard. For example, Mosaic (now Netscape) was originally the primarly used graphics based web page browser, and was distributed freely.[194] Netscape Navigator is undeniably protected by copyright, yet Microsoft was able to create a competing web page browser that quickly challenged Netscape's market position, perhaps unfairly.[195] Now the two products are competing standards, and computer users are the beneficiary of the advances the two companies have made.

An examination of the different browsers shows how a judge might apply the factors if there was a copyright infringement suit. For example, a comparison of Netscape with Lynx, a text based browser, shows that the ability to highlight links to other web pages is functionally required in order to implement a World Wide Web browser. In fact, it is required by the HTML programming language itself. Thus, the functional need to compete factor would allow reuse.

More abstractly, Netscape implemented a menu called "Bookmarks" where users could save frequently accessed web pages. Users might create a huge number of bookmarked pages. Reuse of this element leads to two interesting results.

First, Microsoft could argue that it was functionally required to allow users to save bookmarks in its program. In fact, it did create a "Favorites" menu. Note that the non-protection of this "standard" element did not lead to stagnation; rather, in order to gain additional sales, Microsoft improved on the concept by allowing for an easily organized hierarchy of the favorites menu where the use could group specific types of web pages by category. Netscape followed suit by also implementing this improvement.

193. Sun Microsystems, Inc. v Microsoft Corp., 21 F. Supp.2d 1109 (N.D. Cal. 1998) (issuing preliminary injunction).

194. Boyle, *supra* note 74, at 50.

195. United States v. Microsoft Corp., 980 F. Supp. 537, 540 (D. D.C. 1997) *rev'd,* 147 F.3d 935 (D.C. Cir. 1998) (issuing injunction against bundling Internet Explorer with Windows 95).

Second, Microsoft could have created a "bookmark" translation program that would convert Netscape Bookmarks into Microsoft Favorites. Microsoft would have based this program on the customer need for compatibility factor. The amount of time that it would take for a customer to switch all of his or her bookmarks might preclude a person from switching to Microsoft's web page browser, despite the fact that Microsoft's browser was free.

The reuse of certain elements, however, did not have an effect on all of the market choices made by the two companies, nor did it lead to a wholesale substitute product. For example, Netscape's product integrates World Wide Web browsing with an electronic mail/usenet news program. On the other hand, Microsoft chose to implement the same features as two separate products, which could be, and in fact are, used individually if the user chooses.

In addition, from its first release, Microsoft distributed its program as a reusable ActiveX component, and many software companies implemented or embedded, Microsoft's web browsing components within their products (such as Intuit's Quicken). On the other hand, Netscape has not made its browsing features available for others to use, though it recently announced that it was making its source code available for all to see, implying that others could freely modify the source to embed Netscape functions in third party programs.

This illustration is useful because it points out that while a company might protect its product as a whole, as Microsoft or Netscape can, different companies will make different choices about how much reuse to actually allow based on its own private benefits and costs. Given the model and factors above, it appears that eventually the same outcome is achieved, whether by court intervention (such as *Lotus v. Borland*) or by acquiescence based on likely judicial outcomes (such as the bookmarks v. favorites).[196]

VII. CONCLUSION

Copyright protection for computer software will be more important in the coming years as more software is created and sold. The model presented here describes how courts have acted in the past, and the decision factors attempt to guide future courts without being over or under protective. If the decision factors are used, courts may continue to make efficient copyright protection decisions, and at the same time remain consistent in their statements of law. Furthermore, software authors should be able to protect their investment while at the same time have a clearer picture of what prior building blocks they may reuse.

196. Boyle, *supra* note 74, at 49 (noting that whether the outcome is beneficial will be an empirical question).