Rethinking Software Tying

Renato Mariotti†

This Note proposes a new method of product definition in software tying cases. “Tying” is a refusal to sell one product unless the buyer also purchases another product and is a method by which firms are thought to be able to use their power over one product to obtain power over another.

Tying is the principal antitrust allegation brought against Microsoft, which refuses to sell its operating system (Windows) unless the buyer also receives a copy of its web browser (Internet Explorer). Yet Microsoft claims that it has integrated Internet Explorer into Windows, creating a single integrated product. But in order to have tying, two separate products must exist. This dilemma has been dubbed the “single-product problem” and has required courts to define products before conducting further analysis.

This Note explains why traditional methods of product definition cannot be applied to the software industry. The software industry not only requires a new method of defining products, however; it also, by its very nature, transforms the role of the court in software tying cases. After outlining a theory of software evolution, this Note presents a new method of evaluating integrated software and applies it to three recent examples, including the current case against Microsoft. The new test reveals that the combination of Internet Explorer and Windows should not be considered a single integrated product, because the bundling harms consumer welfare.

Introduction ................................................................. 368
I. The Traditional Method of Product Definition and Its Limitations .................................................. 370
   A. Tying Precedents ......................................................... 370
   B. The Inapplicability of the Jefferson Parish Test ......... 374
      1. The “Market Practices” Test ..................................... 374
      2. The “Consumer Perceptions” Test .......................... 377
II. The Nature of the Software Industry Transforms the Role of the Courts in Software Tying Cases .......... 379

† J.D., expected May 2001, Yale Law School; A.B., 1998, The University of Chicago. I would like to thank James Boyle, Larry Lessig, and George Priest for their advice and encouragement. I would also like to thank Michael Fransella, Max Minzner, Nathan Newman, and Kim Roosevelt for their comments and suggestions. This Note was originally written in March 1999, although some effort has been made to update the Note to include developments since that time.

Copyright © 2000 by Yale Journal on Regulation
Introduction

The software industry was born when IBM, under the pressure of four antitrust suits which alleged “tying,” decided to halt its practice of bundling free software with its computer systems.¹ Now this same sort of antitrust challenge is being used against Microsoft, the dominant software company. “Tying” is a method by which firms use their power over one product to obtain power over another, and the Supreme Court has defined a tie as “an agreement by a party to sell one product but only on the condition that the buyer also purchases a different (or tied) product.”² At first glance, the current antitrust challenge against Microsoft seems to fit the Court’s definition quite well. After all, the principal allegation against Microsoft is that consumers cannot purchase Microsoft’s operating system³ (Windows) without also receiving Microsoft’s web browser⁴.

³ In a case before the D.C. Circuit, the court defined an operating system as “the central nervous system of the computer, controlling the computer’s interaction with peripherals such as keyboards and printers.” United States v. Microsoft Corp., 147 F.3d 935, 938 (D.C. Cir. 1998) [hereinafter Microsoft II]. I claim that it is difficult to create any single definition of operating systems. See infra notes 73-82 and accompanying text.
⁴ The court, in Microsoft II, defined a web browser as a software program that “enable[s] users to navigate the [World Wide] Web and to access information.” Microsoft II, 147 F. 3d at 939.
Rethinking Software Tying

(Internet Explorer). Thus, it seems that Microsoft may be using its monopoly power in operating systems to increase its market share in web browsers.

Yet Microsoft claims that it has integrated Internet Explorer into Windows. If Windows and Internet Explorer are regarded as a single integrated product, not two separate products bundled together, then Microsoft cannot possibly be tying the sale of two products. Thus, if we categorically regard all integrated software as a single product for antitrust purposes, all software tying claims would be precluded. On the other hand, if operating systems and web browsers are seen as rigid, fixed product categories, Microsoft is most likely engaging in anti-competitive behavior. Although there are other conditions that must be met in order for a tying arrangement to exist, the question of whether software integration is a form of tying rests chiefly upon our definition of products.

Yet software products resist easy definition, rendering traditional methods of product definition obsolete. It should come as no surprise that commentators have reached simplistic and wildly implausible conclusions when applying existing tying doctrine to the current case against Microsoft. For example, one recent article concluded that there is "little doubt that Internet Explorer and Windows 95 are distinct products under the antitrust law of tying," while another article, published around the same time, came to the exact opposite conclusion: "Microsoft can evade liability under a tying approach . . . simply by integrating its Explorer browser into its Windows operating system." Unfortunately, it is not that simple. A closer look at the software industry reveals products that are extraordinarily mutable, requiring a new approach that does not merely observe integration, but also evaluates it.

In this Note, I abandon the traditional approach to product definition and present a theory of the software industry that profoundly affects the appropriateness of traditional methods of defining software products in antitrust tying cases. I then use this model of the software industry to formulate an alternate method for testing software tying claims, one that evaluates the impact of the integration on consumer welfare instead of

5 The Supreme Court has identified three conditions which, taken together, would establish a per se violation of the Sherman Act. First, there must be two distinct products. See Jefferson Parish Hosp. Dist. No. 2 v. Hyde, 466 U.S. 2, 21 (1984). Second, the seller must possess market power in the "tying" product. Third, in the market for the "tied" product, a "not insubstantial" volume of commerce must be affected. See, e.g., Fortner Enters. v. United States Steel Corp., 394 U.S. 495, 501 (1969). In antitrust parlance, the "tying" product is the product over which the firm possesses market power (e.g., Windows). In a tying arrangement, the firm uses its power of over the "tying" product to gain power over the "tied" product (e.g., Internet Explorer).


relying upon observations of market practices or consumer perceptions. In Part I, I outline traditional methods of evaluating tying cases and explain why they cannot be applied correctly to the software industry. In Part II, I present a theory of the software industry, arguing that the nature of software transforms the role of the court in software tying cases. In Part III, I examine two proposals for analyzing software tying cases that explicitly evaluate the benefits of the new, integrated product. In Part IV, I outline a new approach to adjudicating software tying claims. In Part V, I apply this test to three cases: memory-management software, Netscape Navigator 2.0, and the current antitrust case against Microsoft. Finally, I conclude that Microsoft should be precluded from integrating Internet Explorer into Windows.

I. The Traditional Method of Product Definition and Its Limitations

A. Tying Precedents

Until the Supreme Court's extensive discussion of product definition in _Jefferson Parish_, it had never provided a clear method for defining products in tying cases. The Court explicitly addressed the issue in three cases, but never set forth criteria that could be universally applied. Thus, no standard test emerged. When the Court first considered the single-product problem, in _Times-Picayune_, it held that advertising inserts in morning and evening newspapers owned by the same publisher were a single product. The Court based its analysis solely upon the perception of consumers, which in this case were advertisers. Next, in _Jerrold Electronics_, the Court affirmed per curiam a district court decision which held that "a community television antenna system cannot properly be characterized as a single product." The district court stated four criteria that tended to show that the antenna system was not a single product. First, Jerrold's competitors offered all the equipment necessary for a complete antenna system but did not sell their equipment exclusively as a single

8 Netscape is the name of the software company founded by the creator of the first graphical web browser. See Direct Testimony of Jim Barksdale at 31, United States v. Microsoft Corp., 1998-2 Trade Cas. (CCH) ¶ 72,261 (D.D.C. 1998) [hereinafter Barksdale Testimony], available at <http://www.usdoj.gov/atr/cases/ms_testimony.htm>. Barksdale is the Chief Executive Officer of Netscape and testified on behalf of the government. Netscape's Navigator and Communicator programs have been popular web browsers since their inception.
9 466 U.S. at 18-25.
10 _Times-Picayune Publ'g Co. v. United States_, 345 U.S. 594 (1953).
11 _See id._ at 613.
13 _Id._ at 559.
Rethinking Software Tying

package; second, the number of pieces in each system varied, so that no
two versions of the alleged product were the same; third, the customer was
charged for each item, rather than in a lump sum for the entire system; and
fourth, although Jerrold sold cable antennas which it did not manufacture,
it only required that the electronic equipment in the system be bought from
it. Finally, in *Fortner*, 15 the Court held, with little analysis, that
prefabricated buildings offered by one part of a corporation were a
separate product from loans that were offered by another part of the same
corporation. 16

The lack of a clear standard led lower courts to adopt a patchwork of
inconsistent tests. Some courts decided that separate products existed with
little or no analysis, 17 some used a variant of the *Jerrold* test, 18 and others
looked to market practices as a guide. 19 The Ninth Circuit developed its
own distinct test, which focused on the “function of the aggregation” and
considered “whether the amalgamation of products resulted in cost savings
apart from those reductions in sales expenses and the like normally
attendant upon any tie-in, and whether the items are normally sold or used
as a unit with fixed proportions.” 20 This test was used in a number of
important Ninth Circuit decisions. 21 At least one court in another
jurisdiction tried to combine the Ninth Circuit test with analysis employed
by other jurisdictions to create a single, comprehensive test. 22

The Supreme Court ended this era of conflicting methods of analysis
with its decision in *Jefferson Parish*, which contained its most extensive
discussion of product definition. In *Jefferson Parish*, an anesthesiologist
brought an action against a hospital, charging that it tied hospital services
and anesthesiological services. The hospital argued, conversely, that it
“merely provided a functionally integrated package of services.” 23 The
Court’s analysis of the single-product issue rested upon whether offering
the items separately would, given consumer demand, be efficient: “[N]o

14  See id.
16  See id. at 507.
Cir. 1971); Dehydrating Process Co. v. A.O. Smith Corp., 292 F.2d 653 (1st Cir. 1961).
388 F.2d 272 (2d Cir. 1967); Associated Press v. Taft-Ingalls Corp., 340 F.2d 753 (6th Cir. 1965).
20  Siegel v. Chicken Delight, Inc., 448 F.2d 43, 48 (9th Cir. 1971).
21  See, e.g., Hirsh v. Martindale-Hubbell, Inc., 674 F.2d 1343, 1347 (9th Cir. 1982); Moore
(D. Mass. 1979) (articulating a four-step test which examines the product structure, scrutinizes the
defendant’s product-marketing practices, looks at market practices, and analyzes the efficiencies
gained by the product combination).
tying arrangement can exist unless there is a sufficient demand for the purchase of anesthesiological services separate from hospital services to identify a distinct product market in which it is efficient to offer anesthesiological services separately from hospital services."²⁴

The Court inferred the efficiency of bundling and the nature of consumer demand indirectly from readily observed facts such as market practices, concluding that "other hospitals often permit anesthesiological services to be purchased separately."²⁵ In fact, as the Court noted, it was much more common for anesthesiological services to be unbundled from hospital services: "As a statistical matter, only 27 per cent of anesthesiologists have financial relationships with hospitals."²⁶

Jefferson Parish's "market practices" test parallels Professor Areeda's "primary test" of product definition, which looks at competitive markets to determine whether there are one or two products: "[C]ourts adjudicating the separate products issue look to readily observable facts about whether comparable competitive markets provide the items bundled or unbundled."²⁷ It also has been used in several lower court opinions²⁸ and was cited and upheld in Eastman Kodak,²⁹ the latest tying case heard by the Supreme Court. In Eastman Kodak, the Court looked at market practices to determine that service and replacement parts for Kodak photocopier equipment are two distinct products: "Evidence on the record indicates that service and parts have been sold separately in the past and still are sold separately to self-service equipment owners."³⁰

It is also possible to read the product-definition test in Jefferson Parish in a different light, however.³¹ Focusing on the Court's statement that "the answer to the question whether one or two products are involved turns not on the functional relation between them, but rather on the character of the demand for the two items,"³² the test could become whether the arrangement links two distinct product markets "that were

²⁴ Id. at 21-22.
²⁵ Id. at 23.
²⁶ Id.
²⁹ "For service and parts to be considered two distinct products, there must be sufficient consumer demand so that it is efficient for a firm to provide service separately from parts." Eastman Kodak Co. v. Image Technical Servs., 504 U.S. 451, 462 (1992).
³⁰ Id. at 462.
³¹ I think that this is an inferior reading of Jefferson Parish, but I will examine both possible interpretations in this Note.
distinguishable in the eyes of buyers.\textsuperscript{33} Thus, the fact that patients \textit{perceived} that anesthesiological services were distinct from hospital services would have demonstrated that they were indeed separate products.\textsuperscript{34}

Although the current antitrust suit against Microsoft is not the first tying claim that involves software,\textsuperscript{35} it is the first tying case that has been brought against a manufacturer for combining two software products.\textsuperscript{36} Yet it is not the first action brought against Microsoft by the Department of Justice. The Justice Department's initial dispute with Microsoft ended in a consent decree between Microsoft and the government.\textsuperscript{37} Although the decree was rejected by the district court, the D.C. Circuit reversed that decision.\textsuperscript{38} Two years later, the Antitrust Division of the Department of Justice filed a petition for civil contempt, arguing that Microsoft had violated the terms of the consent decree by requiring original equipment manufacturers to license and pre-install Internet Explorer 3.0 as a condition of licensing Windows 95. Although the district court once again ruled against Microsoft,\textsuperscript{39} it was later reversed by the D.C. Circuit.\textsuperscript{40} On May 18, 1998, before \textit{Microsoft II} had been decided, the Justice Department brought the current antitrust suit against Microsoft,\textsuperscript{41} alleging violations of the Sherman Act, including tying. Many have compared \textit{Microsoft III} to \textit{Standard Oil},\textsuperscript{42} predicting that it will be "one of the most

\textsuperscript{33} Id.

\textsuperscript{34} Testimony that patients and their physicians frequently do differentiate between hospital services and anesthesiological services and request specific anesthesiologists, was provided by Dr. Roux, and other anesthesiologists as well. There was no testimony that patients or their surgeons do not differentiate between anesthesiological services and hospital services when making purchasing decisions.

\textsuperscript{35} See Service & Training, Inc. \textit{v.} Data Gen. Corp., 963 F.2d 680 (4th Cir. 1992) (overturning the district court's decision that diagnostic software and repair software are inextricably bound together); Virtual Maintenance, Inc. \textit{v.} Prime Computer, Inc., 957 F.2d 1318 (6th Cir. 1992) (finding that software support and hardware maintenance were separate products); Datagate, Inc. \textit{v.} Hewlett-Packard Co., 941 F.2d 864 (9th Cir. 1991) (determining that software maintenance and hardware maintenance were separate products); Digidyne Corp. \textit{v.} Data Gen. Corp., 734 F.2d 1536 (9th Cir. 1984) (finding that central processing units and operating systems were separate products); Data Gen. Corp. \textit{v.} Grumman Sys. Support Corp., 761 F. Supp. 185 (D. Mass. 1991) (ruling that diagnostic software and computer equipment repair services were separate products).

\textsuperscript{36} For the purposes of this discussion, I use the term "software tying" only to describe cases in which the sale of one software product is tied to the sale of another software product.


\textsuperscript{38} See United States \textit{v.} Microsoft Corp., 56 F.3d 1448 (D.C. Cir. 1995) [hereinafter \textit{Microsoft III}].


\textsuperscript{40} See Microsoft II, 147 F.3d 935 (D.C. Cir. 1998).

\textsuperscript{41} See United States \textit{v.} Microsoft Corp., 1998-2 Trade Cas. (CCH) ¶ 72,261, at 82,671-72 (D.D.C. 1998) ("Microsoft III").

\textsuperscript{42} Standard Oil Co. \textit{v.} United States, 221 U.S. 1 (1911).
contentious battles in business history\textsuperscript{43} and will "write the rules for competition in the Information Age."\textsuperscript{44}

B. The Inapplicability of the Jefferson Parish Test

1. The "Market Practices" Test

Due to the high concentration of market power in a handful of firms, courts cannot rely on market practices in software integration cases. In \textit{Jefferson Parish}, there were thousands of hospitals with which one could compare the hospitals in the Jefferson Parish Hospital District. In contrast, software products almost always have, at most, one or two serious competitors. The strong tendency of software markets to be dominated by one large firm is due to a phenomenon called "network effects." Network effects arise in cases in which "the utility that a user derives from consumption of the good increases with the number of other agents consuming the good."\textsuperscript{45} For example, as more people purchased VHS video recorders, other owners of VHS machines benefited from an increased selection of recorded material in the VHS format, while owners of Beta machines saw the value of their equipment decline, as fewer stores made recordings available in the Beta format.

Software is often viewed as a paradigm case of a product that has strong network effects.\textsuperscript{46} Network effects are easy to identify in the case of operating systems, for example, because application software—such as games, word processors, and web browsers—will usually only function with one particular operating system. Thus, software designers must decide which operating system to write for, and they usually pick the most popular operating system. This results in "a 'positive feedback' process in which more and more applications are written for a dominant operating system such as Windows."\textsuperscript{47} Needless to say, this process makes Windows even more attractive to consumers because of its compatibility with an ever-increasing number of applications.\textsuperscript{48}


\textsuperscript{45} Michael L. Katz \& Carl Shapiro, \textit{Network Externalities, Competition, and Compatibility}, 75 AM. ECON. REV. 424, 424 (1985). In \textit{Microsoft II} the court provided its own explanation of this phenomenon: "[A]n increase in the number of users of a particular item of software increases the number of other people with whom any user can share work . . . thereby increas[ing] the value of [the product] to consumers." \textit{Microsoft II}, 147 F.3d at 939.

\textsuperscript{46} See, e.g., Mark A. Lemley \& David McGowan, \textit{Legal Implications of Network Economic Effects}, 86 CAL. L. REV. 479, 491 (1998); Katz \& Shapiro, supra note 45, at 424.

\textsuperscript{47} Piraino, supra note 7, at 17.

\textsuperscript{48} As Nobel laureate Kenneth Arrow put it, "[t]he value of the operating system product is
concentrated. From 1993 to 1997, Microsoft and its closest competitor controlled a combined 95% of the spreadsheet market, 88% of the word processing market, and about 79% of the market for database programs. From 1993 to 1997, Microsoft and its closest competitor controlled a combined 95% of the spreadsheet market, 88% of the word processing market, and about 79% of the market for database programs. Currently most users obtain this software as part of a bundle of several programs, including a word processor, a spreadsheet, and a database manager. Microsoft’s Office Suite now controls 93% of the market for these lucrative software bundles.

Because software markets are usually dominated by one or two products, the “market practices” test is inapplicable to the software industry. In a software market, it is impossible to compare the practices of one firm with market power to the larger competitive market as a whole because there is usually only one real competitor. Applying the “market practices” approach to the current case against Microsoft, for example, would force us to evaluate the efficiency of Microsoft’s integration of its web browser into its operating system based solely upon the practices of marginal competitors such as BeOS, which is installed on a mere 20,000 personal computers worldwide. The absurdity of this situation is demonstrated by Microsoft’s testimony in the ongoing trial that the practices of competitors such as Linux and BeOS justify its integration of Internet Explorer into Windows. The chief executive of Be Inc., the company which produces BeOS, was rightfully outraged. As he put it, “Microsoft has a lot of nerve using my tiny company as an example.”

The head of Red Hat Software, which makes the most popular version of Linux, also pointed out the absurdity of the comparison: “It just tells you how desperate Microsoft is for a competitor that they’re holding up a software box produced by 100 guys in the hills of North Carolina. Who are they trying to kid?”

Trying to apply the “market practices” approach to the web browser market yields equally bizarre results. Because Microsoft and Netscape

(last modified Jan. 7, 1999) <http://www.neramicrosoft.com/level_1/nera_an.html>. It is important to note that this paper was sponsored by Microsoft, and that I am citing market share data from the paper because it is the only source available. Obtaining market share data directly from the source, International Data Corporation (IDC), costs thousands of dollars per report. I have no reason to believe, however, that Evans et al. altered the IDC data in any way.

55 See id.
56 See Joel Brinkley, Microsoft Has a Stronghold in Office Suites, N.Y. TIMES, May 27, 1998, at D1.
57 See Joel Brinkley, Footnotes in the U.S. v. Microsoft, N.Y. TIMES, Feb. 15, 1999, at Cl.
58 See Schmalensee Testimony, supra note 49, at 114 ("To ensure that users have the software to implement Internet-related features, all current versions of the operating systems of which I am aware include Internet-client software, either in the form of a separate browser that is shipped with the operating system or as an integrated part of the operating system itself.").
59 See Brinkley, supra note 57 (quoting Jean-Louis Gassee).
Rethinking Software Tying

operating systems, it is important to consumers that there are over 25,000 applications available for Windows and only 258 applications available for Red Hat Linux. It is easy to understand, therefore, why Microsoft controls ninety-five percent of the PC operating system market. The end result of this self-perpetuating process is that a de facto standard is created. As the head of one software company put it, "[t]he Windows operating system is to most computer users and providers of software applications or Internet services what the dial tone is to businesses and customers using the telephone: It's the thing you must have access to if you are going to communicate with each other."

Network effects are not limited to operating systems: De facto standards are created in nearly every category of software. As more consumers purchase a particular software product—Microsoft Word, for example—"it becomes easier for each previous user to share files without the need for a conversion program, and easier for employees to switch jobs without retraining." The high cost of switching programs "locks in" many users, who are reluctant to learn a new program and convert their existing files to the new format. The result is that every software market is dominated by a handful of programs. For example, market-share data suggest that from 1981 to 1997, the markets for word processors, spreadsheets, and database programs were dominated by the top three firms in each respective market, with an average combined share of between 80% and 90%. Recently, these markets have grown even more

in its capability to run application software. The larger the installed base of a particular OS, the more likely it is that independent software vendors will write programs that run on the OS, and, in this circular fashion, the more valuable the OS will be to customers." Declaration of Kenneth J. Arrow, United States v. Microsoft Corp., 980 F. Supp. 537 (D.D.C. 1997) (No. 94-1564), quoted in Direct Testimony of Frederick R. Warren-Boulton at ¶ 50 n.26, Microsoft III, 1998-2 Trade Cas. (CCH) ¶ 72,261 (D.D.C. 1998) [hereinafter Warren-Boulton Testimony], available at <http://www.usdoj.gov/atr/cases/ms_testimony.htm>


51 Direct Testimony of William Harris at 1-2, Microsoft III, 1998-2 Trade Cas. (CCH) ¶ 72,261, available at <http://www.usdoj.gov/atr/cases/ms_testimony.htm>. Harris is the President and Chief Executive Officer of Intuit Inc. and testified on behalf of the government.

52 Lemley & McGowan, supra note 46, at 491.

53 See Drahter, supra note 1, at 706 ("This investment of time and energy in training and development has locked me into WordPerfect 5.1 as surely as if I had signed a pledge in blood to be its loyal user forevermore. I would no more think of converting to Microsoft Word or some other new word processing program, and thereby losing all my investment in macros and training, than I would consider doing all of my professional writing henceforth in Swahili.").

54 See David S. Evans et al., The Rise and Fall of Leaders in Personal Computer Software
control a combined 97% of the market for web browser software, the only significant "market practices" will be the practices of those two companies. Deciding whether the practices of one company are anti-competitive based solely upon the practices of their only competitor makes very little sense. Taken to its logical extreme, this would prevent Microsoft and Netscape from adding new features to their web browsers unless they both added the same feature simultaneously.

It might still be possible to use the "market practices" test if the software industry were regionally fragmented. Courts often compare market practices in different regions in order to determine what competitive market practices are. In Jefferson Parish, for example, the Court's use of national data allowed the Court to compare the arrangement in East Jefferson Hospital to competitive markets in the rest of the nation. This sort of comparison cannot be made for software, however, because software markets are global.

2. The "Consumer Perceptions" Test

Consumer perceptions are also a poor standard to use when defining software products. Using consumer perceptions to define products in any industry is problematic because whether or not a consumer believes that something is a single, integrated product is based upon factors such as the product's appearance and the way in which it is advertised. Indeed, these are precisely the sort of factors that the Justice Department urged the court to consider in Microsoft II. Jim Barksdale, the chief executive of Netscape, testified in the Microsoft II trial that Internet Explorer is "undisputedly" a separate product because that is the way it is marketed by Microsoft and covered by the press: "If you look on almost any major web site, you will see a 'download Internet Explorer' button—not a download an 'Upgrade to Windows' button. Microsoft also has participated in the 'browser wars' . . . comparing the features of Internet Explorer to Netscape Navigator or Communicator." Justice Department economist Frederick

61 See Warren-Boulton Testimony, supra note 48, ¶ 142. Warren-Boulton is an employee of Microeconomic Consulting and Research Associates, Inc. (MiCRA) and testified on behalf of the government. Microsoft's data also show that the current combined market share of Microsoft Internet Explorer and Netscape Navigator is 97%. See Schmalensee Testimony, supra note 49, at 155.


63 See, e.g., Alex Y. Seita, The Role of Market Forces in Transnational Violence, 60 ALB. L. REV. 635, 635-36 (1997) ("For commodities such as sound recordings, movies, computer hardware and software, motor vehicles, clothing, and agricultural products, there is essentially a global market.").

64 "[The Department] says that . . . Microsoft provides [Internet Explorer] separately to end users, sells versions of IE 4 for different operating systems, advertises IE 4, tracks its performance in a 'browser market,' and distributes it on a separate CD-ROM." Microsoft II, 147 F.3d 935, 946 (D.C. Cir. 1998).

65 Barksdale Testimony, supra note 8, at 50.
R. Warren-Boulton has made a similar argument, stressing that "browsers were identified as separate products with their own brand names."\(^{66}\) Presumably, based on this logic, if Microsoft marketed Internet Explorer as an "upgrade to Windows," one could conclude that they were not separate products. In this way, the consumer perceptions standard provides an incentive for Microsoft to run lots of advertisements which portray the Windows/Internet Explorer bundle as a single product. Legal standards should not be based upon advertising campaigns.

Further, when applied to software products, the consumer perceptions test loses what analytical value it may enjoy more generally. Unlike most consumer products, software is extraordinarily mutable and its appearance can be altered very easily. As the court noted in Microsoft II, "[s]oftware code by its nature is susceptible to division and combination in a way that physical products are not . . . .\(^{67}\) This property of software, which I will hereafter refer to as "mutability," enables Microsoft to tailor the appearance of its products in order to shape consumer preferences. The fact that the web browser in Windows 98 appears to be seamlessly intermeshed with the rest of the operating system, while the web browser in Windows 95 looks more separate, undoubtedly affects whether consumers regard the products as separate or integrated. For example, the new integrated look of Internet Explorer in Windows 98 was enough to convince one Wall Street Journal columnist that they are a single product, even though it contains almost exactly the same features as the previous version of Internet Explorer when installed on a computer running Windows 95.\(^{68}\)

I agree that Internet Explorer was once really a separate application program, like Netscape Navigator. In fact, about a year ago I wrote a column that the Justice Department loved, attacking Microsoft's claim that the browser had been fully integrated into Windows 95. I said that, in Windows 95, Microsoft was like a cosmetics company that had taped together bottles of shampoo and conditioner and sold them as a unit, but hadn't produced a combined shampoo and conditioner in a single bottle.

In Windows 98, however, the browser truly has become integrated. That integration is real to users even if a professor somewhere is able to write a special program that removes the browsing capability, as has been claimed at the trial. You could also invent a home centrifuge to separate the shampoo from conditioner in a combination product.\(^{69}\)

\(^{66}\) Warren-Boulton Testimony, supra note 48, ¶ 79.

\(^{67}\) Microsoft II, 147 F.3d at 951.


\(^{69}\) Walter S. Mossberg, Microsoft Combatants Should Keep in Mind the Average PC User,
Rethinking Software Tying

Conducting product definition through consumer perceptions, therefore, does little more than reward Microsoft for blending the appearances of its products. If one accepts the consumer perception test, the inevitable conclusion is that the question of product definition is determined by the way in which the product is treated by the manufacturer. This focus on style, not substance, tells us nothing about the products themselves, particularly since the appearance of software can be easily altered. Therefore, consumer perceptions, like market practices, should not be used as a method of product definition in software tying cases.

II. The Nature of the Software Industry Transforms the Role of the Courts in Software Tying Cases

A. The Mutability of Software Categories

In software cases, it is often implicitly assumed that there are “natural” categories of software, or, at the very least, categories which remain stable over time. This assumption manifests itself in the short, precise definitions of software product categories offered in judicial opinions. For example, Microsoft I defined “operating system” and Microsoft II defined “operating system” and “graphical user interface.” Moreover, in Microsoft III, there has been much debate on cross-examination about the definition of software categories. On one occasion, Justice Department attorney Denise De Mory quoted the 1997 edition of the Microsoft Press Computer Dictionary in her cross-examination of a friendly witness. To demonstrate that Internet Explorer cannot be an integrated part of Windows, she “first read the entry for Internet Explorer, which described the product as a ‘Web browser’ and referred the reader to the definition for that term. She then flipped over to that entry, which defined a browser as an ‘application.’”70 Some even believe that software integration can be defended in terms of the “natural,” “traditional,” or “proper” function of a particular software category. In an oft-quoted PC Week column, Jim Louderback explained why he supported the integration of Internet Explorer into Windows:

The Web browser really should be part of the operating system. What does an operating system do? Well, back in stand-alone days it provided access to resources, including files, printers, backup units, keyboards and monitors. The OS delivered a standard interface for

applications that used those resources. Even classic old OSes such as DOS, CP/M and Unix provided a rudimentary way for users, as well as applications, to access those resources. Remember those arcane commands, like LS, COPY and DELTREE?

Those commands, which essentially ran small applications, blossomed over time into graphical environments such as Windows, Desqview and Motif. These windowed environments included their own file management utilities that supplanted the arcane command-line programs of the past.

When we started connecting these systems up to networks, those utilities gave us ways to manage files stored on servers and other computers as if they were local. And finally, some OSes included generic file viewers that allowed users to browse the contents of those files as if they had the creating application installed.

Then came the Internet, which is essentially a huge network with zillions of HTML-formatted files, accessible to everyone. Our OSes already can access and view multiple file types on local and remote systems. So why is it such a big deal for Microsoft to update its file explorer technology to browse and display the contents of HTML files? Is this any really [sic] different from browsing and displaying text files on a network? Not to me. 71

This discussion of the traditional features of operating systems ignores the fact that the function and scope of operating systems has evolved over time. This is not unique to operating systems—boundaries between software product categories are fuzzy and fluid, often overlapping and changing. The nature of individual product categories often changes dramatically from one version of software to the next. “Operating systems” are traditionally considered the paradigmatic example of a definable software category, yet even they are mutable and have changed over time. 72

The traditional definition of “operating system” is nonsensical. Two recent law journal articles use slightly different formulations of this definition: One calls an operating system “the conduit between the
Rethinking Software Tying

hardware and the application program,” 73 while the other dubs the operating system “the software interface between a computer and application software.” 74 Since many programs which are not considered operating systems would fit within these definitions, they are unquestionably inadequate. For example, BIOS and memory-management software are both conduits between hardware and application programs, yet neither is considered an operating system. 75 Moreover, this definition fails to cover Windows 3.0, which is generally regarded as an operating system but is not a conduit between hardware and application programs. 76 It is rather a DOS shell, or, in other words, a conduit between an operating system and application programs.

The definition of “operating system” used in Microsoft II, while more broad and well-crafted than the traditional definition, is still inadequate. The court defined the operating system as “the central nervous system of the computer, controlling the computer’s interaction with peripherals such as keyboards and printers.” 77 This definition runs into the same problem as the traditional definition of operating systems: Other software products (such as memory-management programs) can fit under this definition as well. Moreover, the best analogue for the central nervous system is the computer’s motherboard—not the operating system. The motherboard physically connects the computer’s “brain,” the CPU, to the other components of the system, such as printers, disk drives, and modems.

Creating any single definition of “operating system” is particularly difficult given that what we think of as an “operating system” has changed dramatically over time. For example, within a six-year span (1990 to 1995) the dominant operating system for Intel-based PCs went from MS-DOS 3.3 to Windows 95. Windows 95 not only dwarfs DOS 3.3 in size, complexity, and features, but it also has a completely different interface. Windows 95 has as much in common with the last version of Microsoft Word as it does with DOS 3.3. Windows 95 and DOS 3.3 are far more different than they are similar: The operating system was utterly transformed in that six-year span. In addition, more transformation may be on the way. New conceptions of “operating systems” could radically change the way that computers function, as well as the way in which we interact with them. For example, many industry observers believe that a web browser plus the Java programming language is the operating system of the future. In Microsoft II, the Court mentioned this hypothesis:

74 Jones & Turner, supra note 72, at 356.
76 See Microsoft II, 147 F.3d 935, 945 (D.C. Cir. 1998).
77 Id. at 938.
“Browsers also have the potential to serve as user interfaces and as platforms for applications . . . providing some of the traditional functions of an operating system.”

Even word processors, which have been perhaps the most stable software product category over the past fifteen years, have changed considerably over time. For example, many programs have been integrated into word processors:

Word processors have a history of swallowing other types of software instead of being swallowed themselves. Whole categories of software disappeared from the market when word processors began to incorporate their features. Stand-alone spelling checkers, envelope addressers, outliners, and on-the-fly abbreviation expanders all sold well until word processors started doing the same jobs better.

Indeed, over the past few years word processors have taken over many of the functions traditionally performed by desktop publishing programs. Word processors have also taken over the simple graphing functions which were once handled by stand-alone graphing products. For example, a program called Microsoft Graph is integrated into Microsoft Word.

B. A Theory of the Software Industry

In the previous section, I established that, in the software industry, fixed product categories do not exist over an extended period of time. This does not mean, however, that we are unable to delineate software product categories at a particular point in time. For example, it is possible to say that at this moment there is a market for “web browsers” and that the major firms in that market are Microsoft and Netscape. If it were impossible to ascertain the current state of software markets, it would also be impossible to determine whether or not a firm possessed market power over the tying product, and thus tying analysis would be inapplicable to software products. For example, if it were impossible to determine that

78 Id. at 939. For a discussion of the possible effects of this new “operating system” on Microsoft’s monopoly, see Piraino, supra note 7, at 10.
81 See Michael J. Miller, Microsoft Office Professional, PC MAG., Feb. 8, 1994, at 199, 199; Michael J. Miller et al., Suite Peeks, PC MAG., Nov. 19, 1996, at 102, 112.
82 For a definition of “tying” and “tied” products, see supra note 5. In antitrust parlance, the “tying” product is the product over which the firm possesses market power. Needless to say, the
there is a market for PC operating systems, one could not conclude that
Microsoft possesses market power in the operating systems market.\footnote{83}

Although software product categories are not ever-changing and are
stable at a given point in time, they are still extremely mutable and can
change dramatically. Over the long term, software products evolve in a
manner similar to the evolution of species in the “punctuated equilibrium”
theory popularized by Stephen Jay Gould. The theory predicts that a
lineage’s history will include “long periods of morphologic stability,
punctuated here and there by rapid events of speciation.”\footnote{84} This picture of
relatively long periods of stability, punctuated by relatively short periods
of significant change, is a useful representation of change within the
software industry.\footnote{85}

The nature of software evolution—in which products undergo drastic
change at certain points in time, but are stable otherwise—is a result of the
way in which software programs are altered. Instead of constantly
updating programs, software manufacturers issue periodic “upgrades”
which contain improvements to the previous version of the program.

For example, the first version of Windows was introduced in
November 1985. The next update, Windows 2.0, was released in
December 1987. Windows 3.0 was not released until May 1990.\footnote{86} Because
the previous version of the software does not degrade or decompose in any
way, manufacturers must add new features and substantial improvements
in order to induce users to purchase the new version of the product. Often,
however, these changes do not alter the nature of the product category. For
example, Microsoft Word 97 contains some features that Word 95 lacks,
but it does not fundamentally alter the nature of the program or add major
new functionality.\footnote{87} On the other hand, product categories can occasionally
be transformed by a new version of a software product. Windows 95 is an
example of a software product which transformed “operating systems” by
integrating two previously distinct product categories—operating systems
(MS-DOS 6.22) and graphical user interfaces (Windows 3.11).

But why does the decision of a single software manufacturer to
requirement that the firm possess market power in the market for the “tying” product is essential—
otherwise, the market could solve these problems on its own.

\footnote{83} The question of how one determines whether or not a firm possesses market power in a
given market is beyond the scope of this discussion. It is merely important to note that tying analysis
relies upon the ability to determine whether a firm possesses market power.

\footnote{84} Niles Eldredge & Stephen Jay Gould, \textit{Punctuated Equilibria: An Alternative to Phyletic

\footnote{85} Obviously, the mechanism of change in software has nothing to do with the evolution of
new species—software lacks things like mutations and subpopulations.

\footnote{86} \textit{See The History of Windows, PC MAG., Aug. 1998, at 105, 109, 110, 114; see also Ken
Polsos, Chronology of Events in the History of Microcomputers <http://www.islandnet.com/~kpolsos/comphist> (cataloging key dates in the history of
microcomputers).

\footnote{87} For example, in Word 97 a cartoonish avatar is now the front-end of Word’s help feature.
integrate a new feature into a product transform our understanding of what that product is? The answer lies within network effects, which magnify the power of a market leader to transform a category of products. Because software markets are usually dominated by one or two competitors, a decision by the leader to change its product (e.g., Windows), can transform an entire category of products (e.g., operating systems). Furthermore, network effects dictate that competitors will mimic the interfaces and features of the market leader. This was the underlying issue behind *Lotus v. Borland*, a copyright case in which Borland copied the command hierarchy from Lotus’s spreadsheet program—which was, when the case was filed, the market leader—so that users of Lotus’s 1-2-3 could more easily switch to Borland’s Quattro Pro. Judge Boudin, in a much-heralded concurring opinion, grounded Borland’s right to copy Lotus’s command structure on the network effects created by Lotus’s interface: “A new menu may be a creative work, but over time its importance may come to reside more in the investment that has been made by users in learning the menu and in building their own mini-programs—macros—in reliance upon the menu.” Soon after a market leader adds a feature or changes its interface, the other competitor or competitors will soon follow. I do not mean to suggest, however, that software products are the only products that can be transformed. If someone added wings and a jet engine to an automobile, it would no longer be the same product it once was. But the combination of network effects, periodic upgrades, ease of transformation, and mutability has a profound effect on the evolution of software products. Moreover, this sort of transformative change is commonplace in the software industry, and the mutability of software makes it difficult to detect these changes.

Figure 1 is an adaptation of a sketch used by Gould to illustrate punctuated equilibrium. The chart represents the evolution of a few selected product categories from 1990-1995. During this short period, new product categories (such as data compression software) were introduced, some product categories (such as text-based windowing environments) were eliminated, and other categories (such as operating systems) were transformed. These changes were not gradual, but occurred at particular “points of great change.” These points corresponded to the release of a new version of software that contains new features that transform the nature of the product category.

---

89 Id. at 819.
By definition, software tying occurs at the precise point at which products are transformed—a so-called "point of great change." Presumably, one software product is being integrated into another, and the tying product possesses market power in a sizeable market. Moreover, there must also exist a significant market for the tied product—one cannot imagine the Justice Department bringing a tying case against Microsoft to prevent it from integrating into Windows a product that no one wants. Understanding the precise time at which software tying occurs informs our understanding of the courts' role in software tying cases.

C. An Inescapable New Role for the Court

Many observers have recoiled from the notion that courts could possibly prevent Microsoft from integrating Internet Explorer into Windows. They fear that courts will begin meddling in software design. As Microsoft Chief Technology Officer Nathan Myhrvold put it, "[t]he specter here is of a Federal Bureau of Operating Systems." Yet they fail to mention the implications of inaction by the courts. Allowing the

---

integration of Internet Explorer into Windows cannot be compared to merely allowing a firm to bundle salt with its salt-manufacturing machines. Rather, it is akin to permitting the firm to combine them into a new product which would most likely render both salt and salt-manufacturing machines obsolete. In short, allowing integration is as much a design decision as preventing integration.

For example, if the courts allowed Microsoft to integrate speech-recognition software into its popular word processing program, one could not possibly measure the impact of the integration by looking at the markets for word processors and speech-recognition software, for in the near future those markets would no longer exist. The courts' decisions, in large part, determine the sort of markets we will have in the future. The word processing program, as we now know it, would be transformed forever, and the stand-alone speech-recognition program would likely become a thing of the past. In this way, even by deciding not to interfere with the integration of one software product into another, the court is expressing a preference for a particular product design. 91

Thus, in software tying cases, courts are not merely deciding how products can be marketed: They are determining how products should be designed. Courts cannot escape this responsibility; they can only confront it. Most antitrust scholars, however, advocate running away. Professor Areeda, for example, argues strongly for complete deference to the manufacturer in all software tying cases:

> [Courts] lack the technical expertise to judge product design. As a result, allowing tying inquiry to proceed is likely to result in errors that would deter socially desirable innovations and variations in product design. To screen out such overdetering tying inquiry, courts can and should find a single product. 92

Some, like Judge Easterbrook, go even further, arguing that deference should be the default rule in all antitrust cases: "[J]udicial errors that tolerate baleful practices are self-correcting while erroneous condemnations are not." 93

The argument for judicial deference was also made by the Court of Appeals in *Microsoft II*. Writing for a 2-1 majority, Judge Williams concluded that "the limited competence of courts to evaluate high-tech

91 Judges who defer to the manufacturer would object to the notion that they are constraining future product design, and as a formal matter they are not. But the example of the word processor with integrated speech recognition illustrates why, in practice, allowing a monopolist to integrate can effectively constrain product design.
92 10 AREEDA ET AL., supra note 27, ¶ 1746, at 226.
product designs and the high cost of error should make them wary of second-guessing the claimed benefits of a particular design decision.\textsuperscript{94} For this reason, he argued, "[a] court's definition of a claim of integration must be narrow and deferential."\textsuperscript{95} The two-pronged test he used in the case certainly fits that description.

The first prong of the \textit{Microsoft II} test is that "the combination offered by the manufacturer must be different from what the purchaser could create from the separate products on his own."\textsuperscript{96} This criterion is intended to ensure that the integrated product has "a degree of unity, something beyond merely placing disks in the same box."\textsuperscript{97} While this standard does prevent manufacturers from calling two "disks in the same box" an integrated product, it does little else. As the majority noted, "[s]oftware code by its nature is susceptible to division and combination in a way that physical products are not."\textsuperscript{98} The ease with which a programmer can commingle the two products in a way that "the purchaser . . . [could not] on his own"\textsuperscript{99} renders this standard extremely deferential. Essentially, any addition of software code to connect two products would satisfy this low standard.

The second prong of the \textit{Microsoft II} test requires that "[the integrated product] must also be better in some respect; there should be some technological value to integration."\textsuperscript{100} In and of itself, this standard is not fully deferential. After all, one could imagine an integrated software product which is not better than its two component products acquired separately. Yet the majority cuts the legs out from under this requirement when it indicates "that this analysis does not require a court to find that an integrated product is superior to its stand-alone rivals."\textsuperscript{101} It is hard to understand how an integrated product could possibly be "better in some respect" if it is not "superior to its stand-alone rivals." Yet the court backs itself into this corner out of fear that it might "put[] judges and juries in the unwelcome position of designing computers."\textsuperscript{102} Thus, "[t]he question," said the court, "is not whether the integration is a \textit{net} plus but merely

\textsuperscript{94} Microsoft II, 147 F.3d 935, 950 n.13 (D.C. Cir. 1998).
\textsuperscript{95} \textit{Id.} at 949-50.
\textsuperscript{96} \textit{Id.} at 949.
\textsuperscript{97} \textit{Id.} at 948.
\textsuperscript{98} \textit{Id.} at 951.
\textsuperscript{99} \textit{Id.} at 949. For example, Microsoft Office is the name of a bundle of application programs which can also be purchased separately; no one, including Microsoft, claims that it is an integrated product. In fact, Microsoft calls it a "bundle." Yet Office not only includes application programs such as Word or Excel, it also includes a toolbar which makes it easier for users to launch the programs. Thus, Microsoft Office would presumably pass the first half of the \textit{Microsoft II} single-product test—a purchaser could not create the toolbar on her own.
\textsuperscript{100} \textit{Id.}
\textsuperscript{101} \textit{Id.} at 950.
\textsuperscript{102} \textit{Id.} (quoting 9 AREEDA ET AL., \textit{supra} note 27, ¶ 1700, at 15).
whether there is a *plausible* claim that it brings *some* advantage."\(^{103}\)

The court’s account of the “facially plausible benefits” of the integration of Internet Explorer into Windows reveals how hollow this second requirement is.\(^{104}\) First, the court points out that integrating a web browser into the operating system “allows applications to avail themselves of that functionality without starting up a separate browser application.”\(^{105}\) This “facially plausible benefit[]” is essentially nothing more than a reassertion of the claim that Windows and Internet Explorer are a single, integrated product. Obviously, if Internet Explorer is part of the operating system, applications will not have to “start[] up a separate browser application.”\(^{106}\) But this hardly seems like much of a benefit—after all, by definition, any product integrated into Windows would have the advantage of not requiring applications to start up another program. Second, the court claims that “components of IE 3.0 and even more [of] IE 4—especially the HTML reader—provide system services not directly related to Web browsing.”\(^{107}\) This is merely a more specific restatement of its first point. Because Internet Explorer is integrated into the operating system, other programs can use its components—including its HTML reader.\(^ {108}\) If that benefit is enough to allow integration to go forward, however, then Microsoft would be able to integrate *any* program into Windows. As Judge Wald noted in her stinging dissent,

> The majority’s considerable deference to Microsoft’s plausible claims of advantage . . . barely raises the bar of [the consent decree] above ground level. It is difficult to imagine how Microsoft could not conjure up *some* technological advantage for any currently separate software product it wished to “integrate” into the operating system.\(^{109}\)

The necessary outcome of extreme deference is to allow software companies to integrate any two products and force consumers to purchase the new product that results to obtain either. As Judge Wald pointed out, “the majority’s proposal . . . surrenders to any bona fide assertion of a ‘plausible’ benefit of integration,”\(^{110}\) essentially “fashion[ing] a broad exemption from the antitrust laws for operating system design.”\(^{111}\) The ultimate result, she noted, is that “the majority comes close . . . to judicial

---

103 Id. at 950 (second and third emphases added).
104 See id. at 951.
105 Id.
106 Id.
107 Id.
108 I will examine this specific issue more closely in Part IV.
109 Id. at 961 (Wald, J., dissenting).
110 Id. at 958 n.3.
111 Id.
Wald was right: Judicial abdication is not an appropriate or correct response to extremely important cases such as *Microsoft II*. Although mistakes might be made, allowing every possible software tie to go unscrutinized is not the solution. After all, by trying to avoid designing software, a court is essentially endorsing a particular software design. It is far better for the courts to consciously evaluate design than to inadvertently approve one particular design over another by deferring to the manufacturer.

III. Proposed Methods of Evaluating Integrated Software in Tying Cases

A. *Professor Areeda's Suggestion*

Despite Professor Areeda's advocacy of extreme judicial deference, he does casually propose a test which evaluates integrated software products. In the middle of his lengthy treatise, Professor Areeda devotes a short paragraph to this test:

> Finally, consider the bundling into one software package of previously unbundled operating systems and applications software. Is the seller of such a software package immune from tying inquiry on grounds that it reflects new product design? The answer is no if he can show only that his brands of software operate better in conjunction with each other than with other software. To find a new product, *the items of software must operate better when bundled together by the seller than they would if they were distributed on different diskettes and installed by the buyer*.

Essentially, the only way to make this paragraph consistent with Areeda's strong penchant for judicial deference in high-tech cases is to interpret this test to be something akin to the majority's "plausible benefit" test in *Microsoft II*. However, I will, for the moment, consider this test without reference to the rest of Areeda's views.

Essentially, Areeda's test is whether the integrated product "operates better" than the separate products. The problem is that what "operates better" means is a mystery. After all, a program cannot run any "better" than it is intended to run. One can only (a) make the program run faster, or (b) add features to the program. The former is almost completely

---

112 Id. Wald is not the only critic who accuses the majority of abdication. Professor Dorf, for example, considers the decision "a judicial retreat." Michael C. Dorf, *Foreword: The Limits of Socratic Deliberation*, 112 HARV. L. REV. 4, 13 (1998).

113 10 AREEDA ET AL., supra note 27, ¶ 1746, at 229 (emphasis added).
dependent upon the speed of the computer hardware, because the software
code has a relatively small impact on the speed of the program. Moreover,
integrating a new product into another would ordinarily be expected to
decrease speed because the product’s code would become larger and more
complex with the addition of new features.\textsuperscript{114} The second possibility,
adding features to the program, may very well be unrelated to the
integration itself. For example, Windows 98 (which includes Internet
Explorer) includes features that Windows 95 (\textit{sans} Internet Explorer) does
not. However, many of these features do not require the integration of
Internet Explorer and could have been added even if Microsoft had
decided not to combine the two products. The real improvements that
should be considered are those that would not have been possible without
integration.

Moreover, Areeda ignores other important factors which should have
been considered. First, he ignores the cost of the added software to users
who want the tying product but do not want the tied product. The
innovations made possible by integration are not necessarily desired by
consumers. For example, integrating a trigonometry tutorial program into
Windows might allow for new innovations, thereby making Windows
“operate better” in a general sense. But if very few users want to learn
trigonometry on their computers, then this “innovation” is of no benefit to
the vast majority of users. On the contrary, the added cost of this program
(reflected both in the price of Windows and in the size of the program on a
user’s hard drive) would be a detriment to many users. Second, Areeda
also ignores transaction costs.\textsuperscript{115} By combining two products, the
manufacturer may lower the cost of obtaining the products—at a
minimum, a consumer will only have to purchase and install a “single”
program. This benefit, however small, might justify some types of
software integration, but is ignored by Areeda.

B. \textit{Judge Wald’s Test in Her Microsoft II Dissent}

Judge Wald’s test in \textit{Microsoft II} implicitly accepts the courts’ new
role in software integration cases. She correctly begins her analysis by
declaring that consumer welfare should be the standard by which the
integrated product is considered: “[T]he courts must consider whether the
resulting product confers benefits on the consumer that justify a product’s

\textsuperscript{114} See Direct Testimony of John Soyring at 12, Microsoft III, 1998-2 Trade Cas. (CCH) ¶ 72,261
Soyring is an IBM employee and testified on behalf of the government.

\textsuperscript{115} For a more detailed discussion of transaction costs, see infra Part IV.A.
She then outlines a test which balances two factors: synergies and the evidence of distinct markets. To Wald, "synergies" consist of "evidence that there are real benefits to the consumer associated with integrating two software products . . . ." These "synergies" must be balanced with the evidence "that a genuine market exists for the two products provided separately . . . ."

Although Wald is correct to look at the integrated product's benefits to consumers, her use of "the evidence of distinct markets" as a factor in her analysis generates erroneous results. Under Wald's "balancing test," "[t]he greater the evidence of distinct markets, the more of a showing of synergy Microsoft must make in order to justify incorporating what would otherwise be an 'other' product into an 'integrated' whole." On the other hand, "[i]f the evidence of distinct markets is weak, then Microsoft can get by with a fairly modest showing . . . ." The result is that Microsoft (and other firms possessing market power) are only permitted to integrate programs that consumers do not want into highly desired products. If users of Microsoft Windows also desire web browsing software, then Wald's test places a heavy burden on Microsoft. On the other hand, if Windows users do not want web browsing software, Microsoft's burden is markedly lower. The result is that Microsoft's burden is relatively low for products that Windows users do not want, and relatively high for products which they do desire. The goal should be precisely the opposite. We want firms that produce a highly desired product to integrate only products that the users of the original product already use or want. Giving Microsoft free reign to integrate unwanted or unneeded software into Windows certainly does not confer "benefits on the consumer."

Yet the example used by Wald to illustrate this point does seem reasonable at first glance. Wald notes that "Windows 95 includes a built-in calculator program with relatively few functions . . . ." Correctly asserting that "there may be few synergies associated with building this program (Calculator) into the operating system," she concludes that its integration is nonetheless permissible because "there is also not likely to be much of a market for this program provided separately." Wald's assertion that there is "not likely to be much of a market" for calculator

---

116 Microsoft II, 147 F.3d at 958.
117 Id.
118 Id.
119 Id. at 959.
120 Id.
121 I am disregarding, for the moment, "synergies." The point is that Wald sets a lower standard if Microsoft tries to integrate programs that consumers do not want than for programs that they highly desire.
122 Id. at 958.
123 Id. at 958-59.
124 Id. at 959.
programs is empirically false. Before the integration of Calculator into Windows, users purchased a calculator program as part of a package in Central Point Software's popular package PC Tools.125 Moreover, even with the inclusion of this program in Windows, there are a great number of shareware and freeware calculator programs available.126 Their availability indicates interest in the product, despite their low cost. Furthermore, nearly every Windows user needs to use Calculator at some point. Since users need the product, it makes sense to assume that there would be a separate market for Calculator. Thus, Calculator should be integrated into Windows not because there is a low demand for the product, but rather because it is a program that every Windows user needs at some point. The inclusion of other small built-in Windows utilities such as CD Player can be justified in much the same way.

Furthermore, although Wald’s definition of “synergies” is better than the definition used in the majority opinion, it still does not fully take into account the percentage of users who will benefit from the integration. As I discussed earlier, the majority’s requirement that “there should be some technological value to integration” is far too broad and deferential. Wald’s definition of “synergies” is more strict, requiring “evidence that there are real benefits to the consumer associated with integrating two software products.”127 Yet she does not explicitly take into account the fact that “the consumer” is, in fact, more than one person. It is quite possible that an integrated software product might provide “real benefits” for some consumers and no benefits for others. She also fails, therefore, to take into account the additional cost—monetary and otherwise—which will be borne by users who do not need or desire the tied product. Moreover, like Areeda, she ignores the likelihood that integration reduces transaction costs.

In short, Wald’s test is an excellent beginning but is deeply flawed. Yet it should not be easily dismissed—Wald is the first judge involved in a software tying case who tried to free herself from the twin phantoms of tradition and deference. Her ultimate reliance upon the traditional approach and “evidence of distinct markets” should not blind us to her contribution. By grounding her decision on whether the “resulting product confers benefits on the consumer,” she points us down the right path.

---
125 See Schmalensee Testimony, supra note 49, at 119.
126 For example, on April 6, 1999, I searched for the word “calculator” on Download.com and found 161 calculator-like programs available for immediate download. Many of these calculators had specialized applications, such as Loan Calculator Plus 2.1, which allows users to run multiple calculators for side-by-side comparisons. Many of the 161 calculators had been downloaded several thousand times since the inception of Download.com.
127 Microsoft II, 147 F.3d at 958.
IV. A New Method for Testing Software-Integration Claims

A. The Relevant Factors

Keeping in mind the new role of the courts, I will articulate a product-definition test that consciously seeks to promote consumer welfare. Although at first glance it may seem odd to examine whether a combination benefits consumers as a method of defining products, there is no real alternative once we have put aside the traditional methods of product definition. Moreover, since tying is considered a per se antitrust violation, the "single-product" inquiry is often used as a way of injecting rule-of-reason analysis into a tying case. This test merely formalizes that practice. Courts could reach a similar result by never finding that there is a single product—which essentially eliminates the product-definition phase—and then conducting the analysis I outline below as a type of rule-of-reason analysis.

In order to begin our consumer welfare analysis, it is first necessary to know who the consumers of the tying product are. As I pointed out earlier in this Note, it is possible for integration to benefit some consumers and not others. In most instances of software integration, some consumers who use the tying product will not want the tied product. One could imagine, for example, Windows users who do not want a web browser. Thus, any software integration test would have to weigh the interests of one group of consumers against the interests of another. For this reason, the percentage of users of the tying product who also want the tied product is an important factor in my analysis.

The second factor, and the most important possible benefit of integration, is innovation. For the purposes of this analysis, I define innovation in software integration cases to be functionality that could not have been added to the products had they not been integrated. This definition of innovation is a strict one, but it is necessary if we are to

---

128 This focus on consumers is consistent with antitrust doctrine. From the point of view of antitrust law, the survival of individual firms is irrelevant. This philosophy is summed up by the Supreme Court's famous statement that the Sherman Act protects "competition, not competitors." Brown Shoe Co. v. United States, 370 U.S. 294, 320 (1962).
129 See supra notes 113-15 and accompanying text (discussing Areeda's suggestion); supra notes 116-27 and accompanying text (examining Wald's test).
130 Does this definition capture all benefits that we ordinarily think of as derived from a beneficial tie? Objects like a toaster oven do not seem to offer any benefits that meet this strict definition of innovation at first glance. Putting aside the fact that a toaster oven is not, strictly speaking, a tie (because toasters can still be purchased separately from ovens), we can see an obvious benefit from the combination: It saves valuable kitchen space because the consumer does not have to allocate counter space for two separate products. This sort of benefit does not usually exist in the case of software, but if a combination of two software products took up less hard disk space than the two did separately, that would indeed be an "innovation."
distinguish new features inserted by a manufacturer that are incidental to the integration itself. The last clause of the definition is particularly important if we are to put aside features that are incidental to the integration itself. If we take these features into account, an integration could seem beneficial to consumers even if the same benefits could have been available had the two products remained separate. Thus, the features that we must focus on are those that are made possible by the integration itself. For instance, MS-DOS 6.0 was the first Microsoft operating system which included a disk defragmentation program. A user of an earlier version of MS-DOS, for example, had to buy products such as the Norton Utilities or Central Point Software's PC Tools in order to obtain a similar defragmentation program. In and of itself, this is not innovative. But the integration of the defragmentation program into the operating system allowed Microsoft to add functionality that could not have been added otherwise. For example, when a user's computer crashes and the computer reboots, the operating system automatically runs the disk defragmentation program to check for disk errors. Without integration, this function would not have been possible.

Moreover, when weighing the possible benefits of integration we must also take into account how many users are actually benefited by the new functionality. In the case of the integrated defragmentation program, for example, all users benefit since every user will probably want to run a disk defragmentation program in order to increase speed and reduce data loss. One can imagine, however, new functions which would benefit only a fraction of users.

Obviously, innovation of the sort I have identified will not always be easy to spot. In general, I suspect that this factor is the easiest to misjudge or manipulate—after all, I never claimed that judges could ever be good software product evaluators. For this reason, examining possible innovation introduces an inevitable subjectivity to the analysis, because any prediction about what could not occur is speculative. Yet many types of decisions require judges to make their best conjectures about factors that are difficult to ascertain objectively. The difficulty of precisely identifying innovation in this test does not undermine the test as a whole.

Another possible benefit of integration is that it should, by definition, lower transaction costs. Buying separate products instead of a single integrated product involves inevitable "transaction costs" (for lack of a

---

131 See Edward Mendelson, Defragmenters and Multiboot: Organization or Complication?, PC MAG., Sept. 14, 1993, at 154, 154 ("Two of the many features built into DOS 6—but formerly available only through third-party software vendors—are hard disk defragmentation and a menu system that lets you boot into different configurations for different purposes. Defragmentation can save you time and help preserve your data, and multiboot options let you use your system most efficiently."). Note that MS-DOS was not the first operating system to include disk defragmentation, however, a far less popular operating system (DR-DOS) had included this feature earlier.
better term). For example, a user of product $A$ may not realize that he also needs product $B$ and thus does not acquire it and cannot perform tasks he urgently needs to do. Or, at the very least, he may need to make a separate trip to the store to buy product $B$. These sorts of “transaction costs” are admittedly low, particularly in the case of programs such as e-mail readers which are widely available on the Internet. Another reason that transaction costs can be low is that software products are often bundled with other products that the consumer already purchases. For example, most computers are sold with word processing software included. There are some cases, however, when transaction costs can be quite relevant. Perhaps the paradigm case would be if product $B$ is not bundled with product $A$ but is needed if the consumer wishes to use product $A$.

The last factor that must be weighed in my analysis is the cost of the tied product to users who do not want it. The obvious loss to consumers when two products are integrated is in a situation in which a user is forced to buy product $B$, which he does not want, because it is integrated into product $A$, which he needs. Moreover, even if the price of product $B$ is low, there are other costs associated with the unwanted product, such as wasted hard disk space. Thus, taking into account the cost of the product and the percentage of users of the tying product who do not want the bundled product is an important part of this analysis.

B. *The Test*

As I mentioned earlier, this software integration test essentially weighs the interests of one group of consumers against the interests of another. On the one hand, we must take into account the gains in innovation and the reduction in transaction costs which benefit consumers who want the tied product. On the other hand, we must consider the cost of the tied product to consumers who do not want it but do want the tying product. Obviously, the relative weight given to each group of consumers depends on the percentage of the users of the tying product who fall into each category. Thus, my test can be described as a weighing of the reduction in transaction costs and gains in innovation for the percentage of users who want the tied product against the cost of the tied product for the percentage of users who do not want it.\(^\text{132}\)

Due to the difficulty of determining the gain in innovation, I suggest weighing the factors twice. First, one should complete the analysis

---

\(^{132}\) The sort of formula I have in mind is: \(((P \times (T + I)) - (C \times (I - P)))\) where $P$ is the percentage of the users who want the tied product, $T$ is the reduction in transaction costs, $I$ is the gain in innovation, and $C$ is the average total costs associated with the product. I did not express the test as a formula because I did not want to imply that it is possible to weigh these factors with anything approaching a mathematical level of precision.

395
assuming that there is no gain in innovation in order to determine whether or not there is a prima facie case to allow the integration to go forward. Only if integration is not beneficial without the possibility of innovation should the gains in innovation be considered and the factors reexamined.

Some might argue that this test is unnecessary: if a bundle fails this test, it is inefficient and would never be forced on consumers by a manufacturer. If one believes that manufacturers never tie products together when it is inefficient to do so, then every tie will pass this test. We should only insist that courts conduct this analysis if we take seriously, as I do, the possibility that a manufacturer might force an inefficient tie for nefarious purposes. Indeed, if we believe that this possibility does not exist, then tying law serves no purpose and should be nullified. Although some would see this as a positive step forward, I take the existence of tying law as a given and merely seek to limit its application to forced bundles that clearly harm consumers. There is no reason, therefore, that this test could not be applied to non-software products. Because traditional methods of product definition could be used in those cases, however, justifying the use of this approach would require a lengthy analysis that is beyond the bounds of this discussion.

In order to demonstrate how this test can be applied to various software tying cases to achieve correct outcomes, I will now apply the new method I have outlined to three examples, which represent a reasonable cross-section of possible software tying cases.

V. The New Method Applied to Three Software Tying Cases

A. Memory-Management Software

MS-DOS, Microsoft’s popular operating system, dominated the market for PC operating systems from the introduction of the IBM PC in 1981 until Windows 95 took hold in 1995-1996. MS-DOS also happened to only allow programs to use 640 kilobytes of conventional memory. This limitation soon grew too confining, and software that loaded parts of programs into “expanded” and “extended” memory was developed. The market for this type of product, dubbed “memory-management software,” grew tremendously, and programs such as Quarterdeck’s QEMM became best-sellers.

However, beginning in 1991 with the introduction of MS-DOS 5.0, Microsoft began integrating memory management into its operating

133 A “byte” is a basic unit of computer memory.
system software. The EMM386 memory manager in DOS 5.0 did not "do everything . . . QEMM-386 [could] do," but it incorporated the latter program's "two most important" features.\footnote{\textcite{Prosise91}} Shortly after the release of DOS 5.0, \textit{PC Magazine} predicted that "[m]akers of 386 memory managers will take a hard[er] hit."\footnote{\textcite{Prosise91}} Microsoft's next version of MS-DOS added even more memory-management features, including "a serious memory manager."\footnote{\textcite{Ayre91}} The result, said \textit{PC Magazine}, was that "Microsoft's MemMaker/EMM386 combination . . . is more than enough memory manager for most."\footnote{\textcite{Prosise91}} Windows 95 was the last nail in the memory-management market's coffin. Even before Windows 95 was released, \textit{PC Magazine} noted that memory-management software might be "made obsolete when Windows 95 hits the streets next year."\footnote{\textcite{Prosise94}} Its prediction that "memory management products such as NetRoom, QEMM, and 386MAX stand to take an especially hard hit"\footnote{\textcite{Prosise94}} turned out to be an understatement. By 1996, memory-management software was only helpful in "a few special cases" and certainly was not needed "most of the time" by "most users."\footnote{\textcite{Seltzer96}} Indeed, users installing QEMM found that it offered little or no benefit over the memory-management software imbedded within Windows.\footnote{\textcite{Seltzer96}} Needless to say, today the market for memory-management software is substantially diminished.\footnote{\textcite{Cross-Examination}}

Using the software tying test outlined above, we can determine whether or not Microsoft should have been prevented from integrating memory-management software into its operating system. In our first round of analysis, we will ignore possible innovative benefits of integration. The first step, therefore, is to determine how many MS-DOS users also used memory-management software. Nearly every MS-DOS user needed memory-management software to achieve maximum performance, although a minority of users had relatively modest needs and never ran programs which used more than 640k of memory. Making a precise determination is complicated by the changing nature of application software: The number of MS-DOS users who also used memory-

\footnote{\textcite{Prosise91}.}
\footnote{\textcite{Prosise91}.}
\footnote{\textcite{Ayre91}.}
\footnote{\textcite{Prosise91}.}
\footnote{\textcite{Prosise94}.}
\footnote{\textcite{Prosise94}.}
\footnote{\textcite{Seltzer96}.}
\footnote{\textcite{Seltzer96}.}
\footnote{\textcite{Seltzer96}.}
\footnote{\textcite{Cross-Examination}.}
management software rose over time as the memory requirements of software increased. For the purposes of this discussion, let us assume that a majority of MS-DOS users used memory-management software but that there was a small but significant minority who did not.

Now, let us look at the possible benefits of the integration to the majority. As I noted above, we will ignore innovation for now. But transaction cost reductions were uncharacteristically high in this case. Installing software such as QEMM was difficult and time-consuming, and often even experienced users ran into problems.\textsuperscript{144} Integrating memory-management software into DOS made installation much less arduous and, in the case of pre-installed operating systems, eliminated installation altogether. On the other hand, the cost of the integration to the minority was not very great. Memory-management software was inexpensive,\textsuperscript{145} and the price of MS-DOS did not increase significantly as memory management was added.\textsuperscript{146}

Therefore, this integration most likely should have been allowed, even if we do not consider the possible gains in innovation.\textsuperscript{147} A sizeable majority of users received a significant reduction in transaction costs; in exchange, a minority had to bear a relatively low cost.

B. \textit{Netscape Navigator 2.0}

At the start of 1996, Netscape had no serious challengers in the market for web browsers. Netscape claimed in Microsoft III that of the “hits” recorded in January 1996 on web sites it monitored, eighty-seven percent were made using Netscape Navigator.\textsuperscript{148} Microsoft’s survey data showed Netscape with a lower share, but with more than six times as many users as Microsoft had at the time.\textsuperscript{149} Yet web browsers were not the only Internet products surging in popularity at the time—products such as

\begin{itemize}
\item \textsuperscript{144} See Rick Ayre & Laura Cox, \textit{A Guide to Memory Management}, PC \textit{Mag.}, July 1992, at 227, 229.
\item \textsuperscript{147} I do not mean to make a definite judgment in the case of memory management. My goal is rather to demonstrate how the software tying test I outlined would work when applied to an actual case.
\item \textsuperscript{148} Netscape’s survey of Web sites showed Navigator’s share ranging from 85.7% to 87% in the first four months of 1996. See \textit{NETSCAPE COMMUNICATIONS CORP., NAVIGATOR MARKET SHARE—TRENDS} (1996), cited in Schmalensee Testimony, \textit{supra} note 49, at 137.
\item \textsuperscript{149} See Schmalensee Testimony, \textit{supra} note 49, at 137.
\end{itemize}
newsgroup readers and e-mail programs were growing as well. In early 1996, Netscape did not offer products in these markets, which were dominated by other companies such as Qualcomm, creator of the popular Eudora e-mail program. Instead of releasing a stand-alone e-mail program and news reader, Netscape integrated these functions into the newest version of Navigator, version 2.0, which was released in February 1996. At the time, the integration of these products (Netscape Mail and Netscape News) into Navigator posed a long-term threat to companies such as Qualcomm. Should Netscape have been prevented from integrating other products such as Mail and News into its Navigator browser?

First, we must determine what percentage of Netscape users also used e-mail and newsgroup programs. Although I concede that most people who had an Internet connection in February 1996—before the Internet became part of the American mainstream—also used a stand-alone e-mail program such as Eudora, a sizeable minority did not. For example, some users used UNIX programs such as PINE and ELM to read their e-mail directly from a mail server without downloading the e-mail to their hard drive. Moreover, a majority of Netscape users did not use a stand-alone news reader. Even today, the number of people who actively read Internet newsgroups is much smaller than the total number of users on the Internet.

Now let us look at the possible benefits of the integration to Netscape users who also wanted an e-mail program and news reader, ignoring the possible benefits of innovation for now. In this case, the reduction in transaction costs would be nearly zero. The vast majority of Netscape Navigator software is either (a) downloaded via the Internet, (b) distributed with the user's computer, (c) provided with a subscription to an Internet access provider, or (d) provided by the user's employer or school, as are most e-mail programs and most news reader software. Perhaps there is some slight gain for a user if he downloads one huge file from Netscape instead of two smaller files, one from Netscape and the other from Qualcomm, but that gain is quite small. On the other hand,

150 See Netscape Communications Corporation, Netscape Navigator 2.0 (visited May 18, 2000) <http://www.netscape.com/navigator/v2.0/index.html>, cited in Schmalensee Testimony, supra note 49, at 256. One should note that this trend toward integration has continued with Netscape Navigator 3.0 and Communicator 4.0. In 3.0, Netscape added CoolTalk for two-way audio communication, and in 4.0 Netscape included Composer, a basic web page design program, and AOL Instant Messenger, which allows Netscape users to easily communicate with users on America Online.

151 I will bracket the question of whether or not Netscape possessed market power in web browsers, for it is irrelevant to the current discussion. Whether or not Netscape possessed market power, this example illustrates how the new method can be applied.

152 See Schmalensee Testimony, supra note 49, at 195 ("For the July-September 1998 period, I estimate that the roughly 28.7 million copies of Netscape Web-browsing software used as a main-browser were obtained as follows: 6.7 million were downloaded; 6.0 million came with the user's computer; 4.0 million were provided with a subscription to an Internet access provider; 8.5 million were provided by the user's employer or school; and 3.5 million were from an 'other' source (including retail, mail, and gift).")
users who want Netscape but do not want one or both of the tied products must incur some costs. Even if we conclude that Netscape Mail and Netscape News did not contribute to the $49 price of Navigator, these users lost valuable hard disk space and also had to reconfigure Navigator in order to change the default e-mail and news programs, a process which might be confusing to some users. The infinitesimal gains for some users cannot outweigh the small but significant costs borne by others.

Reassessing the case with the benefits of innovation included does not change the result, because Navigator 2.0 had no added functionality resulting from the integration of Mail and News. Although Navigator is connected in some way to Mail and News—for example, when a user clicks on an e-mail link over the web, Navigator launches Mail—it does not work any differently with non-Netscape products such as Eudora. Indeed, Netscape claims that a user who chooses to set Eudora as the "default mail program" does not suffer any lost functionality. Therefore, the integration of Mail and News into Navigator should not have been allowed to proceed.

C. Microsoft III

Let us now apply our new test to the current case against Microsoft. At first glance, this case seems like a no-brainer—after all, do any Windows users not want to access the World Wide Web? Under our method of software tying analysis, if every user of the tying product (in this case, an operating system) also wants the tied product (a web browser) then the integration is beneficial to consumers per se because it merely reduces the transaction costs involved in acquiring two separate products.

However, there are many Windows users who do not want a web browser. Recently, Microsoft allowed Dell Computer Corporation, which has a “close relationship with Microsoft,” to ship Windows “to corporations with the following options: no browser, Microsoft Internet
Explorer, Netscape, or any combination of these items."\textsuperscript{154} While still maintaining that Windows and Internet Explorer are a single, integrated product, Microsoft has allowed Dell to untie Internet Explorer from Windows for "a limited aspect of [its] business with a select group of corporate customers."\textsuperscript{155} Nonetheless, Dell confirmed that many business users do not want a browser included with Windows-based computers that they order. Dell CEO Michael S. Dell said, "[t]here's a large bank that buys computers from us that doesn't want a browser. There's another large bank that buys from us that wants Netscape. There's another large bank that buys from us that wants Internet Explorer. We build machines and load them with whatever software they want."\textsuperscript{156} Another Dell employee claims that some customers do not want web browsers installed on computers they purchase because they wish to "prevent [their] employee[s] from accessing or attempting to access the Internet or World Wide Web."\textsuperscript{157} Companies also might want no web browser on their computers so as to avoid unnecessary computer-resource use.\textsuperscript{158}

Dell is not the only computer manufacturer that sells PCs without a web browser installed. NEC has announced that it is removing Internet Explorer from a new line of laptops in order to respond to the "[number-one] request from . . . corporate customers," who want "to get a plain-vanilla" operating system.\textsuperscript{159} Packard Bell, which is part of NEC, has also noticed that some customers want to have no browsers on their computers in order to prevent "employees potentially being able to get on the Internet or play games."\textsuperscript{160} Furthermore, large corporations are probably not the only consumers who do not want one. It is possible to imagine consumers who do not have Internet access or are purchasing a computer for a particular use only—statistical research, for example—and for this reason do not need or want a web browser. In any event, the fact that some Windows users do not want a web browser complicates our inquiry.

Putting aside innovation during our initial analysis, there is almost no reduction in transaction costs resulting from the integration of Internet Explorer into Windows. Consumers acquire Internet Explorer in the same

\textsuperscript{154} Rajiv Chandrasekaran, \textit{Microsoft Lets Dell Delete Browser Icon; Software Giant Restricts Option for Others}, WASH. POST, Mar. 10, 1999, at E1 (quoting Michael S. Dell, CEO, Dell Computer Corporation).
\textsuperscript{155} Id.
\textsuperscript{156} Id.
\textsuperscript{157} Warren-Boulton Testimony, \textit{supra} note 48, at 38 (quoting Joseph J. Kanicki, Jr., Strategic Commodity Manager, Dell Computer Corporation).
\textsuperscript{158} "An organization may wish to minimize computer resource use (disk space, memory, CPU power, etc.). Unused applications that reside on a computer typically tie up such resources." Weadock Testimony, \textit{supra} note 72, at 16.
\textsuperscript{159} Nick Wingfield, \textit{Packard to Offer Laptops Minus Explorer Icon}, WALL ST. J., June 1, 1998, at B5.
\textsuperscript{160} Warren-Boulton Testimony, \textit{supra} note 48, at 39 (quoting Mel Ransom).
way that they acquire Netscape—largely through Internet downloads and free bundles with the computers that they buy. Thus, there is little or no reduction in transaction costs. Transaction costs are not lowered, for example, if a consumer who would ordinarily receive a web browser bundled with his computer instead receives a web browser which is integrated into the operating system which is bundled with his computer.

Moreover, although Microsoft and Netscape distribute their web browser software for free, the cost of the web browser software to users who do not want it is small but significant. Beside concerns that the browser would "confuse users" or "increase technical-support costs," Internet Explorer also takes up "more than 80 [megabytes] of disk space." Moreover, businesses that do not want web browsers on their computers because of fears that their employees will use them for recreational use must individually delete Internet Explorer from each and every computer they purchase. The costs associated with deleting the program can be sizeable if a corporation purchases thousands of computers. The significant costs incurred by the minority who do not want a web browser outweigh the nonexistent gains received by Windows users who do want a web browser.

Yet one could still approve of Microsoft's integration of Internet Explorer into Windows if significant innovation were made possible by the integration. However, despite Microsoft's claims to the contrary, there is no beneficial innovation created by the integration of Internet Explorer into Windows. Earlier, I defined innovation as functionality that could not have been added to the products had they not been integrated, and distinguished innovation from additional features which are incidental to the integration itself. Because there are no innovative benefits to the integration itself, Microsoft has attempted to focus attention upon features of Internet Explorer which have little or nothing to do with its integration into Windows. Let us examine each of these so-called "innovations" in turn, so that we can understand precisely why none of them fit the definition of "innovation" that I put forward earlier.

First, Microsoft claims that "[b]y breaking the [Internet Explorer] code into separate components, Microsoft made it easier for [software companies] to use those components in their own programs." Microsoft is making two separate claims here: First, it is pointing out that Internet Explorer's code is broken into separate components. There is nothing

161 See supra note 153.
162 See Soyring Testimony, supra note 114, at 10.
163 Chandrasekaran, supra note 154.
165 Schmalensee Testimony, supra note 49, at 112.
remarkable about that—so is Netscape's. Microsoft then goes a step further, asserting that because Internet Explorer code is part of the Windows operating system, it is easier for other companies to use Internet Explorer code when they make their own software products. This argument is meaningless: By definition, any program that is integrated into an operating system will be available for other software products to use. That is, after all, the idea behind an operating system: Application programs are written for a particular operating system, utilizing code made available by the operating system for precisely that purpose. If the fact that Internet Explorer is part of the operating system is reason enough to allow the integration to go forward, then presumably every software product should be integrated into the operating system. More importantly, however, much of this benefit could have been realized without tying the two products. If developers want to write programs using Internet Explorer code, they could do so even if it were possible to purchase Windows without also receiving Internet Explorer. Software developers could even include a copy of Internet Explorer—which is freely available—with their product for those who purchased Windows without Internet Explorer. Although this might mean that fewer consumers might purchase products that make use of Internet Explorer code, this is not problematic: If some consumers would not purchase a product unless they were previously forced to install Internet Explorer, then the overall transaction is a loss for those consumers.

Second, Microsoft argues that the integration allows users to "view information on the Web . . . using the same interfaces[] with which they view information on their own local hard disk[s]." It is odd that Microsoft cites Internet Explorer's common web-like interface as a product innovation. After all, users are not even required to install the interface if they download Internet Explorer: "If you install the optional Windows Desktop Update, IE4 completely replaces the user interface in Windows 95 and NT 4.0." Furthermore, when Microsoft shipped Internet Explorer, it did not automatically activate this feature: "The default setting is the Classic style, which is the original Windows 95 interface. To switch to the Web style interface . . . you have to open the redesigned Folder Options dialog box." Thus, it is hard to understand why Microsoft could not have given the "benefit" of the new Windows interface to users who wanted it without tying Internet Explorer to Windows. As one reviewer in the Wall Street Journal put it, the only difference between downloading Internet Explorer 4.0 in order to use the new web-like interface and having

166 See id.
167 Id. at 113.
168 Bott, supra note 164, at 120 (emphasis added).
169 Id.
Internet Explorer integrated into your operating system is that "this integration is now built in from the start, and unlike Internet Explorer 4.0, it can't really be uninstalled."\textsuperscript{170} Instead of integration, Microsoft could have allowed users who want the new interface to simply download Internet Explorer on their own. Thus, the web-like interface does not meet the strict standards required for a feature to be considered "innovation" under the test I formulated earlier.

Third, Microsoft claims that the integration of Internet Explorer into Windows allowed Microsoft to change the way that Windows Help files are stored. Due to the inclusion of Internet Explorer, "[h]elp files for Windows are now stored in the standard Internet text format, HTML."\textsuperscript{171} There are several reasons why this should also not be considered an "innovation" in this case. First, it is difficult to understand why having help files stored in HTML benefits consumers. To the consumer reading the help file, it doesn't matter which application is viewing the file. All that matters to the consumer are the contents of the file itself. Furthermore, even if we grant that storing help files in HTML is beneficial—it is plausible that using HTML lowers costs for other software companies that create help files—there is no reason why having help files stored in HTML requires one to integrate Internet Explorer into Windows. After all, one can read HTML files using any web browser (such as Netscape Navigator) as well as many word processing programs (such as Microsoft Word). Even if we presume that starting up a web browser or word processor is unwieldy or are concerned about the small percentage of users who do not have a web browser, that would not justify the integration of Internet Explorer into Windows because the same benefit could be achieved if Microsoft integrated a bare-bones HTML reader into Windows instead.

Fourth, Microsoft claims that integration of Internet Explorer into Windows makes its Windows Update feature possible. The Windows Update feature "checks certain Web sites automatically to see if updated versions are available for the Windows components that are in use on the user's system; if they are, the Windows Update feature automatically downloads and installs the updated components."\textsuperscript{172} Even if we assume that it is impossible to create Windows Update in a way which allows it to utilize a non-Microsoft web browser, the Windows Update feature still does not fall under the definition of "innovation" given above. While Windows Update may require having both Internet Explorer and Windows installed, it does not require that the two products be \textit{integrated} in any way. It merely requires that both products be present. Thus, it is hard to


\textsuperscript{171} Schmalensee Testimony, supra note 49, at 113.

\textsuperscript{172} \textit{Id.}
understand how Windows Update could possibly justify tying Internet Explorer to Windows. After all, would we not want to let people who do not want their computer to check websites while they are not present choose not to have a feature like Windows Update installed? Users can presently choose not to install many features that they might not use, such as Multilanguage Support, so why not give them the same choice here? Integration precludes that choice; yet it would be possible for Microsoft to make Windows Update available without integration.

Therefore, in the final analysis, the court should rule against Microsoft on the tying claim in Microsoft III. The integration of Internet Explorer into Windows does not benefit any consumers: There is no innovation created by the integration itself, and any possible reduction in transaction costs is small. On the other hand, the integration potentially harms some users who do not want a web browser. The fact that Microsoft is allowing a favored distributor to untie Windows and Internet Explorer for "a limited aspect of [its] business with a select group of corporate customers" suggests that Microsoft is trying to capture the loss it would incur by forcing these consumers to install a product they do not want as a condition of purchasing Windows while still maintaining that the Windows-Internet Explorer bundle is an "integrated product." Since the integrated product does not benefit consumers as a whole, however, one can assume that Microsoft is tying the two products for anticompetitive purposes. But this is not an easy case, and it is important to note that only a small group of consumers are harmed. While this may require an injunction preventing Microsoft from integrating the two products, it is hardly the basis for a forced breakup of the company.

Conclusion

When the products examined in tying cases were physical in nature—such as steel, automobiles, or chicken—judges could reasonably rely upon consumers or competitors to define the relevant products for them. After all, the typical consumer knows enough to be skeptical of the claims of a company which tapes together a bottle of shampoo and a bottle of conditioner and argues that they are both part of the same, single product. You can see the two bottles, and only a very sophisticated method of combining the two products could escape detection by most consumers.

When the current Microsoft antitrust challenge—the first software tying case—was brought to trial, the single-product problem became far more perplexing. Software presents the court with a dilemma that current methods of product definition are unequipped to handle: It is

173 Evans et al., supra note 54.
extraordinarily mutable, it can be transformed within relatively short periods of time, and it is dominated by a handful of firms in a global marketplace.

In this Note, I have put forward a theory of the software industry and provided a method for defining software products that takes into account the unique features of software. Utilizing a more sophisticated analysis is important, for the stakes in this case are very high. The true impact of any software tying decision will not be borne solely by Microsoft and Netscape: The standards we set in software tying claims will determine the future of the software industry. This is not a single, isolated case. Not only has software integration frequently occurred in the past—as we have seen, Microsoft has integrated numerous features into Windows, such as on-the-fly disk compression and memory management—but software continues to evolve. For example, Microsoft has recently integrated a state-of-the-art media player into Windows. The parallels to the current case are striking: "Just as Netscape once commanded 90% of the browser market, RealNetworks now controls about 85% of the market for media players . . . . And like Netscape’s browser, RealNetworks’s player has the potential to grow into a powerful ‘platform’ for other software . . . ."\(^{174}\) In response, Microsoft has “attack[ed] the hot market for audio and video on the World Wide Web by . . . tightly weav[ing] the media technology into its Windows operating system."\(^{175}\) The choices made by courts now will affect the development of the software industry. Instead of permitting Microsoft to integrate any application into its operating system or breaking the company apart to prevent further integration, we should adopt an approach that examines whether the integration harms consumers. But no matter which way the courts decide in this case, the time has come for courts to embrace the role thrust upon them by technology. If they do not, consumers will suffer the consequences.


\(^{175}\) Id.