IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF COLUMBIA

UNITED STATES OF AMERICA Plaintiff, vs. MICROSOFT CORPORATION Defendant.	Civil Action No. 98-1232 (TPJ)
ATTORNEY GENERAL ELIOT SPITZER, *et al.,* Plaintiffs vs. MICROSOFT CORPORATION, Defendant.	Civil Action No. 98-1233 (TPJ)
MICROSOFT CORPORATION Counterclaim-Plaintiff vs. ELIOIT SPITZER, Attorney General of the State of New York, In his official capacity, *et al.*, Counterclaim-Defendants.	

BRIEF OF PROFESSOR LAWRENCE LESSIG

AS *AMICUS CURIAE*

February 1, 2000

TABLE OF CONTENTS

TABLE OF AUTHORITIES

CASES

OTHER AUTHORITIES

TREATISES

STATEMENT OF INTEREST

At the Court's request, *United States* v. *Microsoft*, Nos. 98-1232, 1233, Order, November 19, 1999, I submit this brief addressing the question of how tying law under Section 1 of the Sherman Act should apply to software products.

I am the Jack N. and Lillian R. Berkman Professor for Entrepreneurial Legal Studies at Harvard Law School. I have been a law professor since 1991, initially at the University of Chicago Law School. I have taught antitrust law at the Yale Law School, and courses related to the law of cyberspace at Harvard, including a seminar devoted to this case. I am presently a Fellow at the Wissenschaftskolleg zu Berlin, in Berlin, Germany.

I have no financial interest in the Defendant in this case; nor do I have any financial interest in any competitor of Microsoft. I own shares in a number of mutual funds. The only stock I own is issued by Marimba Corporation.

The views expressed in this brief are my own. I have drawn upon the advice of a number of colleagues and friends in forming these views. These include Harvard Professor Einer Elhauge, MIT Professors Harold Abelson and Jerome Saltzer, William & Mary Professor Alan Meese, as well as Ben Edelman, Renato Mariotti, Bettina Neuefeind, and Chris Parry.

PRELIMINARY STATEMENT

The government alleges that Microsoft has violated Section 1 of the Sherman Act, by "tying" its "browser product" to various versions of its Windows operating system. Both alleged "products" are software products — products that provide computer functionality through software code. The Court has asked me to address the question of how the law of "tying" applies to an alleged tie of software products.

There is reason to believe that the law in this area is unsettled. While the Supreme Court's most extensive examination of the test for "tying" two products, *Jefferson Parish Hosp. Dist. No. 2* v. *Hyde*, 466 U.S. 2 (1984), has been reaffirmed by the Court, *Eastman Kodak Co.* v. *Image Technical Servs., Inc.,* 504 U.S. 451 (1992), *Jefferson Parish* dealt with a tie between two service products, and *Eastman Kodak* dealt with a tie between a physical product and a service product. The Court has not expressly considered the question of a tie between two software products. Lower courts have, but not without some difficulty. While courts post-*Jefferson Parish* have applied the test to cases raising technological questions,[1] courts considering claims that software products are tied have hesitated before doing so directly. See, e.g., *Caldera, Inc.* v. *Microsoft Corp.,* 72 F. Supp.2d 1295 (D. Utah 1999). The Court of Appeals decision in *United States* v. *Microsoft,* 147 F.3d 935 (D.C. Cir. 1998) [*Microsoft II*], if it can be read to define the law of tying for the Circuit, is just one, if prominent, manifestation of this uncertainty.

This hesitation comes from a reluctance by courts to investigate the intricacies of software design. Despite the fact that in other contexts, courts routinely review design decisions, see, e.g., *Kumho Tire Co., Ltd.* v. *Carmichael*, 119 S.Ct. 1167 (1999) (tire design) and *Castrignano* v. *E.R. Squibb & Sons, Inc.*, 546 A.2d 775, 781-83 (R.I. 1988) (drug design); see also Alan Meese, *Monopoly Bundling in Cyberspace: How Many Products Does Microsoft Sell*, ANTITRUST BULLETIN 106 (Spring 1999), the apparent feeling among a number of courts and commentators is that code is different: that the task of evaluating design decisions involved in technological products is uniquely beyond the ken of federal courts.

---

[1] See cases cited in *United States* v. *Microsoft*, 147 F.3d 935, 960 (D.C. Cir. 1998) (Wald, J, concurring). See also *Data Gen. Corp.* v. *Grumman Sys. Support Corp.*, 36 F.3d 1147 (1st Cir. 1994); *Advanced Computer Servs.* v. *MAI Sys. Corp.,* 845 F. Supp. 356, 367-70 (E.D. Va. 1994).

As a matter of judicial policy, I believe it is a mistake to fetishize code in this way. While I agree that an overly invasive antitrust policy can stifle innovation, I am not a skeptic of courts' ability to understand how software functions; nor do I believe that software technology is so benign that it is advisable for courts to ignore the competitive impact of code-based restraints. Nevertheless, my aim in this brief is to evaluate how the law of the D.C. Circuit applies to ties of software products, and ultimately, how it might apply to the government's Section 1 tying claim in this case.

The answer, in my view, depends upon whether the test articulated by the Court of Appeals in *Microsoft II* is the law applicable to this case. If it is, then it is my view that given the findings of this Court, the government has not made out a claim of tying under Section 1 of the Sherman Act. If *Microsoft II* does not apply, then it is my view that under the best reading of the Supreme Court's tying jurisprudence, the government has made out a claim of tying under Section 1 of the Sherman Act. Whether or not *Microsoft II* applies to this case is a question I consider but offer no opinion about.[2]

RELEVANT FINDINGS

While the first version of Windows 95 offered in the retail channel did not include any browser technology, every version of Windows 95 and Windows 98 offered in the Original Equipment Manufacturer [OEM] channel has had browser technology included in the operating system package. Findings at ¶202. This browser technology has been referred to by Microsoft as "Internet Explorer" [IE], and it has undergone a number of revisions. Internet Explorer 1.0 was included in the first version of Windows 95 supplied to OEMs in July 1995. IE 2.0 was released with OEM Service Release ("OSR") 1 of Windows 95 in November 1995. OSR 2, released in August 1996, included IE 3.0. OSR 2.5, released in September 1997, included IE 4.0. And Windows 98 (second edition), released in March 1999, includes IE 5.0. MS Proposed Findings ¶44 [MS Findings]. At roughly equivalent times, Microsoft also released versions of its IE technology independently of operating system service releases, both to permit earlier versions of its

[2]I also do not discuss (1) whether Microsoft has "appreciable economic power" in the market for operating systems; (2) whether a substantial volume of commerce is affected; (3) whether, assuming the Court finds "two products," they are properly considered tied together; (4) whether Microsoft's alleged tying behavior violates Section 2 of the Sherman Act; (5) whether the Court should adopt a "rule of reason" approach to this case rather than apply the "per se" rule.

operating system to be upgraded to the OSR release level, and to give users of other operating systems similar IE functionality.

Though the name of the browser technology has remained constant, the underlying architecture of IE on the Windows operating system has changed substantially. MS Findings ¶537; ¶161. Versions 1.0 and 2.0 of IE were application programs that were shipped in the package of OEM software denominated "Windows 95." They were not architected to "share code" and could be removed from Windows 95 without affecting its functionality. ¶175. Beginning with IE 3.0, that architecture changed. ¶161, 164. Though the findings do not precisely map the code, it is clear that Microsoft designed these later versions of its browser technology to be more modular. Code providing particular functionality was isolated into modules; these modules could then be "called" by other programs, including the operating system. ¶¶162, 163. The result was that more of the browser functionality was made available to programs on the Windows platform, without those programs having to launch a stand-alone browser running in its own application window. Cf. ¶281.

This architecture has obvious benefits. In particular, it makes it easier for other programs to use or rely upon browser related technologies. Other applications' program developers need not write code to support those technologies. They can instead simply rely upon Microsoft's code. ¶44.

At some point, however, the strategy of this modularized design took on a particular form. Rather than a design that simply isolated browser functionality into one set of modules and operating system functionality in another, the Court has found that Microsoft began building modules that bundled code that provided browsing functionality and operating system functionality in the same file. ¶¶160, 161, 164. The consequence of and, the Court has found, "primary motivation" for this design was that the removal of some modules that provided browser functionality meant the removal of operating system functionality as well. ¶164. This design, while assuring newly exposed Application Program Interfaces [APIs] would be within the Windows OS, prevented users from disabling code that provided browser functionality without also disabling the underlying operating system. At that stage, to remove the code providing browser functionality would be to "break" the operating system.

The evolution of Windows 95/98 can thus be divided analytically into three phases (though the findings are not clear about whether there was ever a pure stage two). In the first, the functionality associated with the browser technology is provided by a single application program. In the second, that functionality is provided by a program that has been modularized, such that other programs can call upon some of the functionality typically associated with browsing technology. And in the third, some of the modules providing browsing technology have been mixed with modules providing non-browsing functionality. The effect, then, of moving from phase one to phase two is that browsing technology is more easily available to other applications. The effect of moving from phase two to phase three is that browsing technology cannot be removed without disabling non-browsing functionality. (In all phases Microsoft required by contract that both products be present.)

Thus while Microsoft's consistent objective since July 1995 has been to assure that its Windows 95 and 98 operating systems include browser functionality, its techniques have changed. At first Microsoft relied upon contract alone to assure that browser functionality was included with Windows; later Microsoft also used the design of its code to assure that browser functionality was included with Windows.

With IE 1.0 and 2.0, OEMs were forbidden from removing the IE application from the suite of programs packaged with the Windows operating system. ¶158. The ultimate consumer was free to remove IE 1.0 and 2.0, as well as other aspects of the OS package. But because these versions of IE were stand-alone applications, removing them removed all the code supporting browser functionality. ¶175.

OEMs were likewise restricted with IE 3.0 and IE 4.0 browser technology. While Windows 95 did provide a technique (the "Add/Remove" utility) for disabling easy access to browser functionality, ¶165, OEMs, this Court has found, were contractually not permitted to use that technique to disable access to IE. ¶176. Executing the "Add/Remove" utility would remove some of the files supporting browsing functionality, but it did not remove all such files. ¶165. In particular, it left files that could be used by other programs to provide browsing functionality. *Id.*

Beginning with Windows 98, the restriction on OEMs was not merely contractual. Microsoft eliminated the technique for disabling browsing technology by not including IE technologies within the

list of functionalities that the "Add/Remove" utility could disable. ¶170. And because the code supporting the OS and browser functionality in Windows 98 was intermixed, it was no longer technologically feasible to disable IE functionality. ¶164. OEMs were therefore constrained both contractually and technologically from disabling IE functionality.

The Court has found three different efficiency costs associated with the move to phase three. First, for consumers who desired no browser functionality, the Court has found that this design will induce "performance degradation, increased risk of incompatibilities, and the introduction of bugs." ¶173. Second, for consumers who want the IE browser functionality, the Court has found the design "unjustifiably jeopardized the stability and security of the operating system." ¶174. And finally, though this cost was not a necessary consequence of moving to phase three, the Court has found that Microsoft hard coded its preference for IE browsing functionality in additional ways: While it has become standard on competing platforms to permit users to select the default browsing technology — whether IE or Netscape Navigator, or another browser technology — the Court has found that Windows 98 does not fully respect that choice. At times, despite a user's choice of another browser as the default, Windows 98 will launch the IE browsing technology to view certain html documents. ¶171. Microsoft did not offer in its proposed findings of fact a justification for this inconsistency, though a justification is conceivable. Nonetheless, this Court has not found any technological justification for this design.

TYING

The Supreme Court has interpreted antitrust law to proscribe certain "tying" contracts since *United Shoe Mach. Corp.* v. *United States,* 258 U.S. 451, 457 (1922). The idea behind this proscription has been that tying contracts are in their nature anticompetitive. If the consumer wanted the tied product, there was no reason to tie it; if the consumer did not want the tied product, there was no reason to permit the seller to "force" the tied product. See *Times-Picayune Publishing Co.* v. *United States*, 345 U.S. 594 (1953).

Early tying jurisprudence was quite undiscriminating in its condemnation of tying contracts.[3] This was a mistake. As courts and scholars have argued, the instances when tying can be said to cause competitive harm are relatively infrequent, and the efficiency gains from tying have often been ignored.[4] Thus while tying contracts have been subject to a form of per se review, skepticism about this per se treatment has been longstanding.

One consequence of this skepticism has been that the "per se" nature of the tying rule is relatively unique.[5] Rather than condemn all tying contracts, the Supreme Court has limited the proscription to those contracts where the seller possesses market power in the tying market, and substantial commerce in the tied market is affected. *Eastman Kodak*, *supra*, 504 U.S., at 461-62.

A significant minority on the Court would go further. As articulated in her concurrence in *Jefferson Parish*, 466 U.S., at 32, Justice O'Connor and three other justices have indicated that they would jettison the per se rule, and condemn tying contracts only after applying the rule of reason.[6]

There is good reason to believe that this skepticism will be relevant in the context of alleged ties of software products. Especially in a newly emerging market, it is sensible to hesitate before condemning a market practice, at least until it has been shown convincingly that that practice is anticompetitive.[7] Historically, antitrust law seems to have gone the other way round — condemning the practice first, and

---

[3] For an account of this early history, see Victor H. Kramer, *The Supreme Court and Tying Arrangements: Antitrust as History,* 69 MINN. L. REV. 1013 (1985).

[4] See, e.g., *Fortner Enters*. v. *United States Steel Corp.,* 394 U.S. 495, 514 n.9 (1969) (White, J., dissenting) (procompetitive justifications); IX PHILLIP E. AREEDA, ANTITRUST LAW ¶1730 (1991); ROBERT BORK, THE ANTITRUST PARADOX 380-81 (1978); HERBERT HOVENKAMP, FEDERAL ANTITRUST POLICY 403-04 (1999); RICHARD A. POSNER, ANTITRUST LAW 171-84 (1976) ("prohibition against ties ought to be radically curtailed"); RICHARD A. POSNER & FRANK H. EASTERBROOK, ANTITRUST 809-10 (2d ed. 1981); Frank H. Easterbrook, *Vertical Arrangements and The Rule of Reason*, 53 ANTITRUST L. J. 135, 143-46 (1984); Benjamin Klein and Lester Saft, *The Law and Economics of Franchise Tying Contracts*, 28 J. L. & ECON. 345 (1985); Alan Meese, *Tying Meets the New Institutional Economics*, 146 U. PENN. L. REV. 1, 59-86 (1997); Keith Wollenberg, Note, *An Economic Analysis of Tie-in Sales: Re-examining the Leverage Theory*, 39 STAN. L. REV. 737 (1987).

[5] See, e.g., HERBERT H. HOVENKAMP, FEDERAL ANTITRUST POLICY 426 (1999) (referring to tying per se rule as "idiosyncratic").

[6] See also Kramer, *supra* note 4, at 1059-62.

[7] Compare Frank H. Easterbrook, *Allocating Antitrust Decisionmaking Tasks*, 76 GEO. L.J. 305, 308 (1987) ("Often it takes a decade or more to determine what a business practice really does.").

only later coming to see that much of what it condemns is in fact competitively benign. But the hesitation of the Court of Appeals, and the importance of the software market, will lead justices of the Supreme Court to think carefully about how the law of tying should be extended to software products.

Nonetheless, the Supreme Court has indicated that it is "far too late in the history of our antitrust jurisprudence to question the proposition that certain tying arrangements pose an unacceptable risk of stifling competition and therefore are unreasonable 'per se.'" *Jefferson Parish*, 466 U.S., at 9. And it is of course not appropriate for this Court to decide this case by second guessing or predicting the Supreme Court's jurisprudence, at least with respect to a statutory matter. See, e.g., *Maislin Industries* v. *Primary Steel,* 497 U.S. 116, 130 (1990). It would nonetheless be prudent for this Court to decide the question about tying software products both under the existing per se analysis and under an alternative analysis. In the context of tying generally, and the tying of software products in particular, it is my view that there is a significant probability that the Supreme Court will modify current doctrine.

The prevailing Supreme Court authority on the law of tying is *Jefferson Parish*, relied upon by the Court in *Eastman Kodak*. As described in *Eastman Kodak*,

> A tying arrangement is "an agreement by a party to sell one product but only on the condition that the buyer also purchases a different (or tied) product." Such an arrangement violates §1 of the Sherman Act if the seller has "appreciable economic power" in the tying product market, and if the arrangement affects a substantial volume of commerce in the tied market.

504 U.S., at 461-62 (citations omitted). In this case, Windows 95/98 is the alleged "tying product"; Internet Explorer is the alleged "tied product." The only question that I will address is whether the functionalities associated with Internet Explorer should be considered a product separate from Windows 95 and 98.

ASSUMING MICROSOFT II CONTROLS

The resolution of these questions turns on the status of the Court of Appeals' decision in *Microsoft II*. That case arose as an appeal from a preliminary injunction by this Court to enforce the terms of a consent decree entered into by the parties in 1994. Microsoft challenged that injunction on a number of grounds, one being the failure to provide adequate notice. *Microsoft II,* 147 F.3d, at 943. The Court of

Appeals agreed, and proceeded to decide whether the government was likely to prevail on remand. This required the court to interpret the 1994 consent decree. It is the Court of Appeals' interpretation of that decree that raises the ambiguity in this case.

At the core of the dispute was the interpretation of Section IV(E)(i) of the 1994 consent decree. That section provided that:

> Microsoft shall not enter into any License Agreement in which the terms of that agreement are expressly or impliedly conditioned upon: the licensing of any other Covered Product, Operating System Software product, or other product (provided, however, that this provision in and of itself shall not be construed to prohibit Microsoft from developing integrated products)….

147 F.3d, at 939. Windows 95 was a "covered product." The government alleged that Microsoft conditioned its license for Windows 95 on an OEM's licensing of an "other product" — Internet Explorer. Microsoft argued that Internet Explorer and Windows 95 were "integrated products" and thus, because of the proviso, exempt from the scope of the decree. 147 F.3d, at 946-48.[8]

The parties to the 1994 decree did not define "integrated products." Nor was the intended purpose of the proviso readily apparent. In the proceedings before me during my brief time as special master in that case, Microsoft did offer a fair reading, in my view, of the purpose of that clause. As Microsoft counsel stated,

> Section IV(E)(i) prohibits the tying of all three categories of products *unless* the claim of tying is based on the creation of an integrated product of which a[n] …"other product" is a component: Although the development (read: creation) of such a product could be conceptualized as a tie, what the proviso tells the reader is that it is *not* to be construed that way for purposes of the Consent Decree—*because the parties agreed that any claims of this nature would only be addressed in a plenary antitrust action under the Sherman Act*.

[8] The proviso exempts "developing integrated products." The Court, and the parties, assumed that "developing" was the same as "developing and licensing," so that if the proviso exempted developing, it also exempted licensing. There is some authority in the European law context, within which the decree was drafted, for distinguishing the two, and reading developing narrowly, see R&D Block Exemption Regulation No.418/85 [1985] O.J. L53/5 as amended on Regulation 151/93 [1993] O.J. L21/8, but the distinction would not matter for the purposes of this case. (I am grateful to Professor Steve Anderman for this point.)

Letter of January 21, 1998, from Richard Urowsky to Lawrence Lessig, 2 (final emphasis added). This understanding is consistent with the government's interpretation of the decree in the 1994 Tunney Act proceedings. In response to a filed comment, the government stated that

> Activity of this sort [i.e., integration] requires case by case analysis, and a broad injunction against such behavior generally would not be consistent with the public interest.

United States v. Microsoft Corporation, Civ. No. 94-1564, Response of the United States to Public Comments, 59 FED REG 59426, 59428 (1994). Accordingly, rather than resolving claims about integrated products in the context of the consent decree, both Microsoft and the government indicated that those claims would be dealt with on a "case by case basis." To the extent the decree was regulating "tying," it was intended, under this reading, to regulate ordinary contract-tying. It was not intended to regulate "integrated products" which "could be conceptualized as a tie."

Thus the proviso was jurisdictional — allocating to the consent decree one class of tying claims, and reserving to a "plenary antitrust action" a different class of tying claims. This difference makes perfect sense of the policies at stake. Consent decrees enable expedited proceedings. They should only regulate behavior that is unambiguously problematic. Ambiguously problematic behavior (sometimes benign, sometimes not) should not be regulated by an expedited proceeding. Such behavior is best left to a context where the nature of the behavior and any anticompetitive threat can be fully explored. The government in the Tunney Act comments indicated that bundling through integration was ambiguously problematic. *Id.* It neither indicated that all such bundling was benign; nor that no such bundling was benign. Rather it said such behavior had to be evaluated on a "case-by-case basis." Where the behavior was ambiguous, then, the decree left it to a "plenary antitrust action."

This background is necessary to understand the meaning of *Microsoft II.* The government, as well as many commentators and at least one court, have criticized the Court of Appeals for a establishing an overly restrictive interpretation of tying law's application to software tying claims.[9] Those criticisms make

[9] See, e.g., Pl. Conclusions of Law, at 57-58 [Pl. CL.]; Michael Woodrow De Vries, *United States* v. *Microsoft*, 14 BERKELEY TECH. L.J. 303, 314-22 (1999); Norman W. Hawker, *Consistently Wrong: The Single Product Issue and the Tying Claims Against Microsoft*, 35 CA. W. L. REV. 1 (1998) (applying *Jefferson Parish* analysis to software

sense, however, only if one reads the Court of Appeals to be establishing the law of software tying for the Circuit. But in light of the interpretation of the decree offered by Microsoft, and the understanding of the proviso offered by the government, there is no reason to interpret the opinion in *Microsoft II* as anything more than an interpretation of this jurisdictional provision. For perfectly sensible reasons, the Court, on this view, gave the term "integrated products" a very broad meaning, thereby assuring that any claims about "integrated products" will be adjudicated in a "plenary antitrust action."

It would not follow on this reading of *Microsoft II,* however, that the Court intended to say anything about how the "single product" test of *Jefferson Parish* should be interpreted in the context of a "plenary antitrust action." Indeed, the Court was quite explicit that "[t]he antitrust question is … distinct. The parties agree that the consent decree does not bar a challenge under the Sherman Act." 147 F.3d, at 950 n.14. Thus, rather than offering a binding interpretation of *Jefferson Parish* as applied to software products — an act that would have been clearly beyond the Court's proper judicial role, given the posture of the case — the Court said that its "task … [was] to discern the bargain that the parties struck." 147 F.3d, at 946. Invoking well established Circuit precedent, the Court wrote that while the consent decree was to be "consistent" with antitrust law, "an antitrust consent decree cannot be read as though its animating spirit were solely the antitrust laws."[10] 147 F.3d, at 946.

United States v. *Microsoft*, 56 F.3d 1448 (D.C. Cir. 1995) [*Microsoft I*], shows precisely the sense in which the decree could be "consistent" with antitrust law, and hence the interpretation of a decree "consistent" with antitrust law, even though the scope of the "integrated products" proviso would not be coextensive with the "single product" test of *Jefferson Parish*. A consent decree is a bargain between the federal prosecutor and a potential defendant, which for reasons of separation of powers need not map an-

---

claims); John E. Lopatka & William H. Page, *Antitrust on Internet Time*, 7 SUP. CT. ECON. REV. 157 (1999); William H. Page & John E. Lopatka, *The Dubious Search for "Integration" in the* Microsoft *Trial*, 31 CONN. L. REV. 1251 (1999); *Caldera, Inc.* v. *Microsoft Corp.,* 72 F. Supp. 2d 1295, 1322-23 (D. Utah 1999).

[10] Microsoft suggests that the Court of Appeals indicated it was deciding the case under tying law because, as Microsoft writes, the Court "referred to the provision of the Consent Decree at issue in that case as an 'anti-tying provision.'" Df. CL., at 3. Actually, what the Court said was that "[b]oth Microsoft and the Department characterize § IV(E) as an 'anti-tying' provision." *Microsoft II*, 147 F.3d, at 946. In the next paragraph, the Court pointed out that "Microsoft is clearly right that the decree does not embody either the entirety of the Sherman Act or even all 'tying'

titrust law generally. *Id.* at 59-60. A defendant does not necessarily concede that the behavior regulated by a decree is illegal; nor does the government, by implication, concede that all behavior not regulated by the decree *is* legal. The decree must be "consistent" with antitrust law in the sense that it cannot exceed the government's prosecutorial power. But the government's prosecutorial power, under constitutional principles of executive authority, is not the same as a court's interpretation of the Sherman Act. There is no authority in the D.C. Circuit for the proposition that a consent decree can be read to imply the contours of antitrust law, and no reason to read an opinion interpreting a consent decree as interpreting the contours of antitrust law either.[11]

It is of course this Court's job to interpret the scope of *Microsoft II*. But when there are two understandings of a higher court's opinion — one that reads it narrowly, as properly applying to the issues raised before it, and sensibly resolving the claims properly presented, and another that reads it broadly, as improperly reaching issues that were not before the court, and adopting a controversial interpretation of federal law that was not even briefed by the parties — charity in interpretation at least should lead a lower court to adopt the first rather than the second reading of that opinion. This seems especially true where the text of the opinion itself expressly indicates its limited reach. 147 F.3d., at 950 n.14.

Nonetheless, a substantial question has been raised by the parties about whether the "framework" used by the Court of Appeals to interpret the word "integrated" in the context of that decree should also be used to decide whether Microsoft's "browser functionality" is a "separate product" for purposes of antitrust law. Microsoft reads the Court of Appeals' decision to so indicate, Microsoft Conclusions of Law at 6 [Df. CL.], and this Court's opinion in the summary judgment stage of this case seemed to indicate likewise. *United States* v. *Microsoft Corp.*, 1998 WL 614485 (D.D.C. 1998), at *10-13. The Court of Appeals' opinion does not speak exclusively of "integrated products" — the language of the consent decree.

---

law under the Act." *Id.* Thus the Court's characterization of Microsoft's position does not seem to support the conclusion that the Court was interpreting tying law generally.

[11] Microsoft argues, "[t]he Court of Appeals thus held that an integrated product constitutes a single product for purposes of antitrust law so long as there are 'facially plausible benefits to [the] integrated design.'" Microsoft Conclusions of Law [Df. CL.], at 6. But I am unsure how the Court of Appeals could "hold" as to a matter that was not before it; nor how it could "hold" anything with respect to "antitrust law" when it had stated expressly that the issues were "distinct."

In a number of places, it uses "integrated products" and "single product" — language drawn from antitrust tying law — interchangeably. E.g., 147 F.3d, at 946. It is thus a reasonable prediction that the Supreme Court would modify existing tying law to fit its proposed standard for determining whether software products should be considered "separate products" or a "single product." For these reasons, it is certainly possible for this Court to view itself bound by the Court of Appeals' opinion, in spirit if not in form. Therefore, for the purposes of this part of my analysis, I will assume that *Microsoft II* set the standard for tying law as it applies to software products generally.

The Court of Appeals said that two software products were "integrated" for purposes of the consent decree if it was more efficient for them to be combined by Microsoft than for them to be combined by OEMs or the consumer. The test had two parts. As the Court wrote,

> We think that an "integrated product" is most reasonably understood as a product that combines functionalities (which may also be marketed separately and operated together) in a way that offers advantages unavailable if the functionalities are bought separately and combined by the purchaser.

147 F.3d, at 947. And later,

> [T]he combination offered by the manufacturer must be different from what the purchaser could create from the separate products on his own.

Id. at 949. This, however, left open the question of when the relevant "combination" occurs — whether at the design or installation stage. If it was the installation stage, then, as this Court has found, ¶190, at least with some versions of Windows 95, there would be essentially no difference between the "advantages []available if the functionalities [were] bought separately and combined by the purchaser," and the advantages available if the combination were made by Microsoft. At that stage, the code from both, and hence functionality of both, would be the same.

The Court of Appeals, however, interpreted the relevant moment of "combination" to be at the design stage of Windows' development, not the installation stage. The relevant "combination," the Court wrote, "is the creation of the design that knits the two together" rather than the stage where the two products are installed. "Consequently," the Court concluded, "it seems clear that there is a reason why the integration must take place at Microsoft's level." 147 F.3d, at 952.

That was the first part of the test for determining whether two products are "integrated." The second part required that the integration actually do some good. As the Court explained, "there should be some technological value to integration." 147 F.3d, at 949.

> The concept of integration should exclude a case where the manufacturer has done nothing more than to metaphorically "bolt" two products together, as would be true if Windows 95 were artificially rigged to crash if IEXPLORER.EXE were deleted.

147 F.3d, at 949. The court was quick to note, however, that "do some good" does not mean "a net plus." 147 F.3d, at 950. The review, the Court instructed, "must be narrow and deferential." *Id.* Rather than asking whether the integration was efficient, or a good idea, or sold more operating systems, the Court said the "question is … whether there is a plausible claim that [the integration] brings some advantages." *Id.*

While the Court relied in its opinion upon the analysis of the Areeda Treatise, its rule is actually more forgiving than the rule announced in the treatise.[12] The question the Areeda Treatise asks, after the plaintiff establishes that two products are "separable," is whether the products *could be* designed such that the consumer could combine them just as well as the defendant. X PHILLIP E. AREEDA, EINER ELHAUGE & HERBERT HOVENKAMP, ANTITRUST LAW ¶1743, at 192-94 (1996) [AREEDA TREATISE]. If there is a reason why the consumer can't bundle them as well as the defendant — for example, if they are too cumbersome, or if the combination would need to occur at an early stage in the product's production — then the "New Product Rationale" that the Court relied upon would treat the two "separable" items as a "single product" for purposes of antitrust tying law. But if there was no reason why the product couldn't be designed to allow the consumer the choice of whether to install the product or not, while keeping the same efficiencies, then the product would be considered "two products" for purposes of antitrust tying law.

The standard in *Microsoft II* is different. The Court does not follow this two step analysis, asking first whether products are "separable" and second, whether there is some good reason that they should,

nonetheless, not be separated. Instead, the court simply asks, consistent with the language of the consent decree, are they "integrated"? Given the product *as designed*, is there a reason why the *defendant* should bundle rather than the consumer? The question the Court addressed was whether they *were* architected to be separate products, not whether they *could be*.

This fact about the Court of Appeals' approach is made clear in the examples it considers. For example, the Court noted that "IE 4 technologies are used to upgrade some aspects of the operating system unrelated to Web browsing," and these features "cannot be included without also including browsing functionality." 147 F.3d, at 951. In saying these features "cannot be included," the Court was not opining about how software could be designed. It certainly understood that Microsoft could have designed its products differently, to permit the IE 4.0 technologies to provide browsing functionality only, while providing users with a separate upgrade package to upgrade operating system functionality. See, e.g., 147 F.3d, at 951. But the Court was not holding Microsoft to the test of what could be designed. The court expressly directed that it was not proposing that "in making this inquiry the court should embark on product design assessment." 147 F.3d, at 949. Because of their "limited competence" courts were not to "second-guess[] the *claimed benefits* of a particular design decision." 147 F.3d, at 950 n.13 (emphasis added). The perspective the Court adopted was to evaluate Windows 95/IE *as they were designed* not as they *could have been designed.*

The standard in *Microsoft II* is different from the test relied upon in the Areeda Treatise in a second way as well. In making its comparison, the baseline the Court adopted was whether the design by Microsoft provided "plausible benefits [] as compared to an operating system combined with a stand-alone browser such as Netscape's Navigator." 147 F.3d, at 950. The issue was no longer whether consumers could combine the stand-alone products (such as IE 4 and Windows 95) and produce the same functionality; the Court had determined that the relevant "combination" was at the design stage. Indeed, the Court expressly rejected the idea that "in installing IE 4 an OEM is combining two stand-alone prod-

---

[12] So, again, if one interprets the Court's opinion as applying antitrust law generally, then its interpretation of the Areeda Treatise at least was mistaken. See Einer Elhauge, *The Court Failed My Test*, Washington Times, A19, July 10, 1998. Professor Elhauge did not address Windows 98.

ucts." 147 F.3d, at 951. "[T]he products," the Court instructed, "do not exist separately." 147 F.3d, at 951-52. Thus the relevant comparison was not whether Windows 98 provides "some benefit" in functionality as compared with Windows 95 plus IE 5.0; the question was whether its benefit compared with "a stand-alone browser such as Netscape's Navigator." 147 F.3d, at 950.

But, again, this is not the methodology of the Areeda test that the Court purported to follow, if the court was interpreting tying law generally. The question the Areeda standard asks is whether combination by the consumer could be as good as the combination by the defendant, not whether the combination by the defendant is as good as a combination with another product. See AREEDA TREATISE, ¶1746 at 229. Thus again, if this Court read the Court of Appeals to be applying the test that the Areeda Treatise summarizes in order to explicate tying law generally, then the Court of Appeals was mistaken.

Between concluding (1) that the Court of Appeals was articulating a new standard for antitrust tying law, but was mistaken in interpreting the doctrine that it purported to rely upon, and (2) concluding that the Court of Appeals was interpreting "integrated products" in the context of a consent decree, using analogous principles from a related area of law, the second would seem a fairer reading for a District Court to adopt than the first. But if the Court believes that though mistaken in its interpretation of the test it was relying upon, the Court of Appeals was setting the standard for antitrust tying law generally, then the methodology of the Court of Appeals is in my view dispositive of the government's Section 1 tying claims in this case, at least as they relate to tying achieved through technological design.[13]

If the relevant "combination" under *Microsoft II* is at the design stage, then there can be no dispute about whether Windows 95/98 and IE pass the first requirement of the Court of Appeals' test — that they are better combined by Microsoft because only Microsoft can design the code. This is true by definition under the test proposed by the Court.

[13] I am therefore not addressing the question whether IE 1.0 and IE 2.0 were improperly "tied" to Windows 95; that bundle was effected through contract alone. ¶155. Nor am I addressing the question whether the vertical restraint on OEMs not to run the Add/Remove program constitutes an improper restraint under the rule of reason. See Alan Meese, *Monopoly Bundling in Cyberspace: How Many Products Does Microsoft Sell*, ANTITRUST BULLETIN 115 & n.168 (Spring 1999).

So too is it clear that the combination passes the second requirement of the Court of Appeals' test —the question whether the integration does "some good," or whether there is a "plausible claim" that the integration "brings some advantages" relative to a "stand-alone browser such as Netscape Navigator." 147 F.3d, at 950. While this Court has found instances of the design of Windows 95/98 that serve no efficiency function — indeed instances that hamper the efficiency of Windows 95/98, ¶¶ 173,174 — the question is not whether there is any bad in the design. The Court of Appeals' test is whether there is "some good." The Court's findings indicate that there is some good, even if the Court believes that on balance, the net is not good.

Nor does the "bolting" qualification that the Court of Appeals articulated change this conclusion. In describing a kind of linking that would fail the test for "integrated products," the Court excepted cases where the designer links products "without the link serving any purpose but an anticompetitive one." 147 F.3d, at 949. But every time the Court articulated this exception, it used language indicating that the anticompetitive purpose must be the *only* purpose. Thus: "without the link serving *any* purpose"; "*nothing more than* … metaphorically 'bolt'"; if the product is not superior "*in some respect*." 147 F.3d, at 949 (emphasis added). The exception for bolting is not an invitation to balance the anticompetitive purpose against other purposes. Rather, in my view, the test requires this Court to accept the bad with the good.

To conclude that the "bolting" that this Court has found was the only effect of Microsoft's design, the Court would have to conclude that the other purported benefits in fact did "no good." But in fact this Court has found some good. ¶¶186, 193 (indirectly recognizing platform value of exposed browser functionality APIs). And while the Court's findings do not address it, it does not seem contested that the modularized design of IE 3.0 and later enabled independent software vendors to write to the APIs exposed by the IE components. Compare ¶44. Nor was it contested — indeed, the government's own witnesses asserted — that this feature provided "some benefit." MS Findings ¶526. Thus under the Court's test, understood as an interpretation of *Jefferson Parish* as applied to software, even if part of Microsoft's motivation were anticompetitive, so long as part of what it produced was beneficial the government's Section 1 tying claim would fail.

It might be argued that at a different level of particularity, Microsoft's design would fail the Court of Appeals' test. While phase two in the three phases of IE's development certainly presents benefits, the government might argue that this Court has found that phase three provided no such benefits. That while there was good reason to expose APIs, and to make browsing technology more available, there was no good reason to hold the operating system hostage to browsing technology by making it impossible to remove browsing technology without also "breaking" Windows. This, the government might argue, was the relevant "bolting." Microsoft has argued that phase three provides other benefits; for example, it assures IE technologies are on all Windows machines, and thereby relieves third-party software vendors of the design and support burden of distributing upgrades to Microsoft's operating system technology. But the Court has not found this to be significant. ¶¶173, 174.

While I do believe that the Court has found that there was no good in moving to phase three of Windows 95/98 design, I do not believe that this approach is consistent with the Court of Appeals' opinion. The Court of Appeals was trying to avoid a situation where courts were "embark[ing] on product design assessment[s]." 147 F.3d, at 949. It would be inconsistent with this objective to imagine courts engaging in lengthy trials about the nature of software design before determining whether at some level of particularity there was "some good." Rather, the Court's language and purpose seem to aim at removing district courts from design evaluations. Thus, in my view, the proper level of generality must be the package of functionality taken as a whole. At this level, under the Court of Appeals test, Microsoft must prevail.

ASSUMING MICROSOFT II DOES NOT CONTROL

If the reasoning of *Microsoft II* does not govern the standards applicable to an alleged tie of software products, then the question is how this Court should apply the standards of *Jefferson Parish* to the facts in this case. In examining that question, I will draw upon the Supreme Court's authority, as well as the understanding of that authority outlined in the treatise begun by the late Professor Philip Areeda. As

both parties in this case, as well as the Court of Appeals and Supreme Court, have made extensive use of that treatise,[14] I take it to be relatively neutral in this dispute.

My analysis is in three parts. I first address the question, "What is a software product?" I then ask whether Windows 95/98 and the browser functionality of IE should be considered "two products" for purposes of antitrust tying law, first assuming the standards of *Jefferson Parish* apply directly to software products, and second assuming those standards require some elaboration.

I. WHAT IS A SOFTWARE PRODUCT?

The essence of a tying claim is that the defendant has improperly linked "two products." A single product cannot be tied to itself; a tie always involves two. But though an alleged tie must always allege the "two products" said to be tied, the conclusion of the tying inquiry may well be that "two products" (or as the Areeda Treatise sometimes calls them, two "items") should be considered a "single product" for purposes of antitrust tying.

But that very formulation raises a question often left implicit in a tying inquiry — what is a "product"?

This question is ordinarily implicit because the answer with physical products is self-evident: with tangible items, a "product" is a physical object that can be sold independently of another "product." As the Areeda Treatise puts it,

> [I]f a car manufacturer insists on selling cars with their tires, we might debate whether cars and tires are a single product. But there would very likely be little dispute about which part of the total contrivance constitutes the "car" and which part constitutes the "tire."

HERBERT H. HOVENKAMP, 1999 SUPPLEMENT TO AREEDA, ET AL., ANTITRUST LAW 493 (1999) [1999 SUPPLEMENT]. The same is generally true for intangible products as well. There is no ambiguity when we refer to the "service contract" that was allegedly tied to the sale of Kodak parts in *Eastman Kodak.* The "product" refers to the set of services that would be provided over the term of a contract to support a particular copier. It plainly doesn't refer, for example, to the particular technicians who would serv-

[14] See, e.g., *Eastman Kodak Co.* v. *Image Tech Servs., Inc.,* 504 U.S. 451, 469, 470 n.15, 474, 476, 499, 491, 493, 497,

ice the machines (even though without these technicians, there would be no service). Nor does it refer to the tools or supplies that would be used in servicing those machines (even though without these, there would be no service). Instead, it refers, abstractly, to the functions performed by the technicians, tools and supplies in maintaining the Kodak copiers.

But as the history of this case reveals, there is a contest about what a "product" is when speaking of a software product. Microsoft insists that because the government cannot point to the "code" that constitutes IE, it cannot identify a "product." MS Findings ¶¶361, 364 ("software products consist of software code and nothing else…"). The government does not point to any particular code that might constitute the product "IE," but it does believe that there is a set of functionalities that count as the relevant "product." Pl. CL., at 59. Without resolving the question, the Court of Appeals appeared skeptical of treating a set of functionalities as a "product," 147 F.3d, at 947, though it expressly indicated that it "did not mean to suggest that [functionality] is never an appropriate criterion of identity." *Id.* As it went on:

> In some contexts it may be appropriate to treat as equivalent two products that supply the same functionality, if they meet the same demand. Computer programs written for different operating systems, however, do not seem to meet the same demand … .

147 F.3d, at 947 n.9.

As this passage indicates, what a "product" is should not turn upon questions of metaphysics. Rather, the definition of a product should depend upon the economic purpose of the inquiry.[15] That purpose, in a tying case, is to identify the product over which the law might seek to assure consumer sovereignty. As the Supreme Court has explained, the aim of antitrust tying doctrine is to assure "that the public, acting through the market's impersonal judgment, shall allocate the Nation's resources and thus direct the course its economic development will take." *Times-Picayune Publishing Co.* v. *United States*, 345 U.S. 594, 605 (1953). This suggests that the perspective from which this Court should identify a software

---

502 (1992); *Microsoft II*, 147 F.3d, at 948 n.11, 949, 950, 951, 962; Df. CL., *passim*; Pl. CL., *passim*.

[15] As the Supreme Court said of the scope of tying law generally, quoting Professor Kenneth Dam, Dam, *Fortner Enterprises* v. *United States Steel: "Neither a Borrower Nor a Lender Be",* 1969 SUP.CT.REV. 1, 19, "the definitional question is hard to separate from the question when tie-ins are harmful. … To treat the definitional question as an abstract inquiry into whether one or two products are involved is thus to compound the weakness of the per se approach." *Jefferson Parish*, 466 U.S., at 21 n.33.

product is the perspective of the *consumer in the market*. If it is the purpose of tying doctrine to preserve consumer choice, it would make most sense to view a "product" as the consumer would. ¶149 ("Consumers determine their software requirements by identifying the functionalities they desire.").

It is for this reason that I believe the government is correct to identify "software products" by their functionality. On this view, a "software product" should be viewed as "functionality separately valued by consumers." This definition focuses on the perspective of the consumer. It is also objective — when defined from this perspective, no party on its own can alter what a "product" is. And while the definition of any "product" will change, the legal test for determining whether two "products" should be considered a "single product" for purposes of antitrust tying law has other ways of accounting for this change. See, e.g., AREEDA TREATISE, ¶1746.

This test is also consistent with the principle articulated by the Court of Appeals. As the Court said, "in some contexts," two "products" would be the same if they met the same demand. 147 F.3d, at 947 n.9. Without the benefit of a factual record, the Court of Appeals assumed that software written for different operating systems could not satisfy the same demand. But in light of the findings that this Court has made, it is plain that computer programs written for different platforms could meet the same demand. The drive to build versions of Internet Explorer for different platforms came from Microsoft's recognition of a significant demand for common "browser functionality." ¶153 (discussing demand). Corporations wanted to standardize on a single browser technology, to minimize the costs of training. ¶151. The aim of Microsoft was to develop versions of Internet Explorer for different platforms that would provide, as much as was possible, the same functionality. This is the same demand even though the programs run on different platforms.

To view "software products" as the code, rather than functionality, not only would not reflect consumer understanding of software, but would create many potential paradoxes of identity. These stem from the nature of software code. There is no necessary relationship between a given set of functionalities and a particular design of code. The same functionalities could be instantiated in very different software. But because the underlying code is invisible to the consumer (at least so long as it is not open source

code), these "differences" would be invisible. The consumer doesn't care, for example, whether the same IE functionality is provided by one file or fifty. It would therefore be meaningless to speak of preserving consumer choice over a particular "product" if the product were the "code." To say one had a choice over such invisible differences would be like wrapping products in gift wrap, and then giving consumers the "choice" of products.

The Court of Appeals hesitated before embracing the notion of a product being identified with its functionality because it found it odd to think about separating or removing a product merely by "hid[ing]" its code. 147 F.3d, at 948 n.11. But if there is any oddness here, it rests in the nature of computers, not in the nature of the government's test for products. It is well known, for example, that when a user "deletes" or "erases" a file, the ordinary technique of operating systems is simply to remove the file from the drive's file listing. The actual contents of the file still remain on the disk, "hidden" from the user, or from the ordinary ways a user gets access to a file, at least until that file is written over by another program. But no one would therefore argue that a user has not complied with an obligation (imposed, say, by a contract) to "remove a file after X days," simply because the user invoked the "delete" function and merely "hid" the contents of the file. The ordinary meaning of "deleting" or "erasing" a file is simply to make it inaccessible; and a file is "deleted" even if a recovery program can reconstruct the file directory entry. Thus while it might be odd to say someone has removed a book from a library simply by removing its card from the card catalog — though many librarians would consider a book "lost" if it is mis-shelved — this is a perfectly ordinary way of speaking about software on a computer. The relevant feature for a consumer is the functionality, not the code, just as "[y]ou don't go out and buy a Frank Sinatra record because you want another piece of plastic in your house. The product is the music." Transcript of Teleconference, United States v. Microsoft, No. 94-1564, January 16, 1998, page 16 (counsel for defendant).

Microsoft resists this understanding of "product" because it says that means that any time functionality is added to its operating system, a tying question is raised. Df. CL., at 5. If indeed that were a consequence of understanding a "software product" as a set of functionalities, then that would be a good reason to resist this understanding of "software product." Operating systems, like applications, have historically folded separate functionalities into a primary product. At one time, no operating system supplied

TCP/IP support; separate products did; now all major operating systems support TCP/IP. Cf. MS Findings, at ¶531. At one time, most word processors had no spell check or grammar check inside; now many do. MS Findings, at ¶147. This addition of functionality is an ordinary part of the evolution of software. And clearly, every such addition should not raise a federal tying claim.

But this is to confuse the initial question of how one defines a "software product" with the ultimate question of whether, for purposes of antitrust tying law, two "items" should be considered a "single product." There is nothing problematic about saying about a physical product that a "car" is a product, and a "radio" is a product, but then concluding that for purposes of antitrust law, a car bundled with a radio should be considered a "single product." The "separate product" or "single product" designation is the conclusion of a legal analysis, not an instance of ordinary language. Like "malice" in defamation law, which has little relation to "malice" in ordinary language, *Masson* v. *New Yorker Magazine,* 501 U.S. 496, 510 (1991), this may be confusing to the ordinary reader. But courts and commentators have little trouble understanding the difference between a description of "two products" going into a tying inquiry, and the conclusion that these two products are a "single product" for purposes of antitrust tying law.

Thus in my view, this Court should define a "software product" as a set of functionalities separately valued by consumers, and identify particular "software products" by looking to the market's ordinary understanding of particular products. See, e.g., ¶150 (defining "web browser"). This would both avoid forcing courts to examine code to understand what a product is, and permit the test to track what is the obvious character of software to most consumers — what it does.

II. ARE WINDOWS 95/98 AND IE BROWSER FUNCTIONALITY A "SINGLE PRODUCT"?

A. APPLYING *JEFFERSON PARISH* DIRECTLY

To say that two "software products" have been bundled into the same software package is not yet to conclude that they should be considered "two products" for purposes of antitrust tying law. That conclusion is the result of a legal analysis that itself is based upon a number of considerations.

The core of the inquiry is whether it is "efficient," as *Jefferson Parish* and *Eastman Kodak* put it, to provide the products separately. *Jefferson Parish*, 466 U.S., at 22; *Eastman Kodak*, 504 U.S., at 462. "Effi-

ciency" is determined indirectly. *Jefferson Parish* does not instruct courts to evaluate the costs and benefits of separating two products. Rather, the aim is to identify proxies for efficiency that are sufficient to indicate that a defendant should be forced to offer two products separately.

The proxy that *Jefferson Parish* fixes on is "separate demand." The essential question is whether "two separate product markets have been linked." 466 U.S., at 21. The evidence *Jefferson Parish* used to find "sufficient demand for the purchase of anesthesiological services separate from hospital services," *id.*, included (1) that the two services could be provided separately; (2) that they were billed for separately; and (3) that doctors or patients often requested specific anesthesiologists, particularly in the specialty at issue in the case. *Id.* at 22-23. These together indicated that "consumers differentiate between anesthesiological services and the other hospital services provided by petitioners." *Id.* at 23.

These factors were not exclusive. The Court pointed, for example, to the opinion of Judge Van Dusen in *United States* v. *Jerrold Electronics Corp.*, 187 F. Supp. 545 (E.D. Pa. 1960), *aff'd*, 365 U.S. 567 (1961), as an indication of the other considerations that might be relevant to the "separate product" inquiry. But these additional factors are simply other ways to answer the same question — whether there is "separate demand." They are not "other factors" in the sense of other considerations that might be balanced against the "separate demand" test. (The Supreme Court, for example, expressly rejected the argument that "functional[] integrat[ion]" should matter to the inquiry. *Jefferson Parish,* 466 U.S., at 19.) These other factors are just other ways to identify whether "two separate product markets have been linked."

This, in my view, is the crucial fact about *Jefferson Parish*. If the Supreme Court really means the "separate demand" test to be the only proxy for whether there is "one product" or "two" for purposes of antitrust tying, and if it really means that the very same test should be applied regardless of the "industry," *id.*, at 26 n.42, then in my view, on the facts that this Court has found, the government has prevailed on the "separate product" question.

The Court has found that browsers can be supplied separately from operating systems; obviously they have and can be billed for separately from operating systems; and this Court has found that many

people make specific requests for browsers, independently from the default selected by an operating system. ¶¶ 150, 151, 153, 154. Consistent with the approach in *Jerrold*, other competitive firms treat the products as separate; none bind the consumer's choice as Microsoft does. These factors are unequivocal to the conclusion that there is "separate demand" for browser technology and operating systems. If *Jefferson Parish* alone is the test, then the government should prevail on the "separate product" question.

But Microsoft argues that a more extensive analysis of the "separate product" test is called for when the products are "software products." I believe that Microsoft is correct. As the 1999 Supplement to the Areeda Treatise notes, "[i]t bears … emphasis that tying law's 'separate product' requirement was not developed with a product such as computer software in mind." 1999 SUPPLEMENT, at 490. Other courts have found it necessary to extend the analysis of *Jefferson Parish* in the context of software products, *Caldera*, *supra*, as have antitrust commentators.[16] The view of these skeptics is that something is missed in an analysis of software products that focuses on "separate demand" alone.

The concern is over-inclusiveness — that the "separate demand" test in the context of software will condemn far too many bundles, especially if the rule is a "per se" rule. As Microsoft argues, the evolution of software is constantly a process of bundling new functionality into old products. As the government acknowledged in the 1994 Tunney Act proceedings, often this bundling involves adding functionality to an operating system that results in the lessening of demand for some software product. 59 FED REG 59426, 59428 (1994). Yet the "separate demand" test places a constant pressure on this bundling. Because the evolution that Microsoft describes is a process where existing products are folded into other products, it will always appear "feasible" to provide the products separately. Yet because software, as the Court of Appeals observed, is "by its nature [] susceptible to division and combination in a way that physical products are not," 147 F.3d, at 951, it is (practically) always true that software *could be* designed to permit these separate products to interact seamlessly. ¶¶162, 163. But if this were the only test, then "virtually all improvements to software would have to be regarded as separate products," 1999

[16] See *supra* note 9.

SUPPLEMENT, ¶1746.1 at 489, and the bundling that Microsoft describes would then be perpetually regulated by the courts.

Some software projects aspire to this "separability." The Open Source or Free Software Movements, both as a matter of programming aesthetic and as a matter of express policy, favor architectures that are modular, and hence easily modifiable. See, e.g., Microsoft "Halloween Document," <http://users.andara.com/~sdinn/halloween.html> ("Linux uses commodity PC hardware and, due to OS modularity, can be run on smaller systems than NT"); Sandra Loosemore, with Richard M. Stallman, Roland McGrath, Andrew Oram, and Ulrich Drepper, *The GNU C Library Reference Manual*, <http://www.gnu.org/manual/glibc-2.0.6/html_chapter/libc_1.html>; Linus Torvalds, *The Linux Edge*, in OPEN SOURCES—VOICES FROM THE OPEN SOURCE REVOLUTION 101 (Chris DiBona et al. eds., 1999) (discussing importance of modularity); CARLISS Y. BALDWIN, KIM B. CLARK, DESIGN RULES: THE POWER OF MODULARITY (1999) (discussing open source movement). But while as a policy matter, one might favor such a design architecture, the antitrust laws have not compelled it. There are many bundles that are benign from the standpoint of consumer welfare, and that antitrust law should therefore permit, even if they are bundles that, under a different design principle, need not have been "bundles."

It is this realization that has led the author of the latest edition of the Areeda Treatise to embrace a very lenient rule for software ties — at least those that bundle software in a new way. As the supplement now suggests,

> [A] single product conclusion seems to be the correct one in all cases in which the code for the two programs is interspersed such that the purchaser cannot readily separate them. The disadvantage of such a rule is that any software producer can comply with it by interspersing code. But the disadvantage of an alternative rule forcing separation is that most of the advantages of integration will have been lost.

1999 SUPPLEMENT, ¶1746.1b, at 494. Needless to say, if this were the rule, then Microsoft would prevail on the "separate product" question.

As a District Court, you must decide this case under existing law. And therefore, if the Court believes that *Microsoft II* does not control, or that, properly read, it was simply an interpretation of the term

"integrated products" in the consent decree, then the Court should conclude under the per se analysis of *Jefferson Parish* that the "separate demand" test has been met.

But in my view, it would be appropriate for this Court to go further. A great deal of energy and social resources have been committed to this case. This Court has had the benefit of an extraordinary clarity and completeness in presentation by the attorneys on both sides. Both sides are too quick, however, to minimize the difficulty of this question of how tying law properly applies to software products. Making the case seem easy is their proper role. But this is not an easy question, and the legal system would benefit greatly from this Court's experience if the Court would, as an alternative, craft a standard that makes sense of the values in antitrust law and of the peculiar facts about software.

In my view, that experience does suggest an approach that avoids invasive antitrust review, and yet does protect against threats to the competitive process. The approach is drawn from the methodology of the Areeda Treatise, though it modifies one aspect of the Areeda test. In the balance of this section, my aim is to sketch this alternative, and its relationship to the findings this Court has made in this case.

B. CONSIDERATIONS IN APPLYING *JEFFERSON PARISH* TO SOFTWARE TIES

My analysis is guided by three considerations that I believe should inform the articulation of any rule governing the tying of software products.

(1) *Risk*: I do not believe software is more benign as a technique for strategic bundling than contract, or "technology" (such as physical products), is. Indeed, as the evidence in this case suggests, software can be a more effective technique for strategic bundling than contract or "technology." This is for two reasons. First, as the Court of Appeals noted, software by its nature is plastic. This means that it is easier to mold software into the form a designer wants than say bridges, or hard disk drives. It also means software is more like "contract" than like "technology" — like lawyers drafting a contract, software designers have great freedom to package software as they wish. ¶¶162, 163. That freedom is less with physical technology. While a designer of a software application could choose to put the functionality of a spreadsheet program into a flight simulator, the designer of an automobile cannot choose to put the functionality of the drive train into a radio.

Second, software is less transparent than other technologies in the bindings that it might effect. Because of the complexity of modern software products, it is often easy to hide the actual functioning of a software routine. Thus, to the extent there is an incentive for strategic behavior, it is easier to hide that behavior in code than it is in either contract or technology.[17] Contracts (since interpreted by humans) and technology (since largely observable) don't keep secrets well. Code — at least code that is not "open source" code — keeps secrets well.

The consequence of both features of software is that the risk that software will be used to effect strategic bundling is greater than the risk that technology generally will be used to effect strategic bundling. Antitrust law therefore does not have less reason to be concerned about strategic ties with software; indeed, these considerations suggest that the law has more reason to be concerned.

(2) *Neutrality:* The test for tying should not itself influence the design of software products.[18] To the extent that software manufacturers seek to bundle software products, or software functionality, they can achieve that bundle either through contract or through the design of the software itself — in short, through code. If the bundle itself is not anticompetitive, then the law should not tilt this design decision in one way or the other. The test for tying should be neutral between contract-based and code-based restrictions on bundling two products.

The reason for this consideration should be obvious. If two software products could be bundled together either through a contract that requires them both to be present if one is, or through software that essentially makes it impossible for one to exist without the other, then there is no evidence in this case, and no argument that I know of, to show why the law should prefer one mode of bundling to the other. While antitrust law is generally encouraging of technology-based tying efficiencies, and skeptical of claims

[17] As the Court of Appeals commented when dismissing the hypothetical bundle of a mouse with an operating system, "[p]roblems seem unlikely to arise with peripherals, because their physical existence makes it easier to identify the act of combination." 147 F.3d, 948 n.11.

[18] This point is analogous to the observation by Meese that hostility to post-transaction tying efficiencies forces inefficient "forward integration." Alan Meese, *Antitrust Balancing in a (Near) Coasean World*, 95 MICH. L. REV. 111, 114 (1996). Page & Lopatka, *supra* note 9, at 1274, make a similar point.

of contract-based tying efficiencies,[19] the bolting achieved through software has no necessary relationship to efficiency. As the Court's findings make clear, code that makes it hard to separate two sets of functionalities need not produce any efficiency. The same "integration" could occur without bolting two products with locked bolts. Or in the terms I outlined in the section on Relevant Findings, there is no strong reason to believe that the move from phase two bundling to phase three bolting produces any increase in efficiency.

But if the law is especially forgiving of one method of bundling over another — if it, for example, scrutinizes bundles achieved through contract more strictly that it scrutinizes bundles achieved through code — then the effect of this rule may be to tilt the architecture of software towards software bundling rather than contract bundling. Unless the law has an independent reason for preferring that technology of bundling, the law of tying has no reason to induce it.

(3) *Screening:* The test for one or two "software products" should avoid, if possible, extensive and direct examination of the efficiencies or benefits of one design over the other. While the government is certainly correct that courts have in other contexts considered the efficiency of various technical and design issues, Pl. CL., at 57 n.11, it is also clear that the objective of the *Jefferson Parish* test is to minimize the factual burden necessary to make the "two product" determination. As Professor Areeda observed, "[t]he separate products requirement is not an invitation to examine the general reasonableness of the bundle," for this

> would thwart the main purpose of the separate products element: *screening* legal inquiries, allowing them to go forward only when the benefits of further inquiry … exceed the costs resulting from the inquiry itself.

AREEDA TREATISE, at 180, 183. The aim instead is to look to readily observable facts about the products and the markets, so as to filter tying cases to avoid extensive judicial inquiry where it is unlikely that an alleged tie does any competitive harm, or where it seems unlikely that judicial inquiry would be socially desirable. Thus tests that require courts to weigh the benefits of a particular software design against the

[19] See Meese, *Bundling*, *supra* note 13, at 80-95.

competitive harm are less preferable to those that allow the "two product" inquiry to be made without extensive examination of the nature of a software design.

C. EXTENDING *JEFFERSON PARISH* TO SOFTWARE TIES

While few commentators or courts have examined tying law as it applies to software products,[20] many have tried to draw from *Jefferson Parish* a useable set of tests to apply to a broad range of products. The Areeda Treatise is a prominent example. Its aim is to articulate the Supreme Court's "separate product" test, in a way that is consistent with its case law, and with as much of the case law from lower courts as possible.

The Treatise organizes the inquiry into two steps. In the first, the plaintiff has the burden of showing "(1) that *some* customers actually want the items separated and (2) that separating them is physically and economically possible." It seems plain from the Court's findings that the government has met this burden. ¶¶150-54. See also AREEDA TREATISE, at 192.

In the second step, the burden of production shifts to the defendant to establish that at least one of six "single product" rationales applies to the facts in the particular case. If any of these tests yield the conclusion that two items ought to be considered a "single product," then the result of the inquiry generally is that these two items be considered a "single product."

Of the tests, or "rationales," that the Areeda Treatise describes, three are directly relevant to the facts of this case.[21] These are (1) the "Competitive Market Practices" rationale, (2) the "New Product" rationale, and (3) the "Same Product" rationale. As will be clear, the focus of the Court's analysis needs to be upon the "New Product" rationale, but the insight that guides that analysis will be found in the "Same Product" rationale.

---

[20] See, e.g., De Vries, *supra* note 9; Rachel V. Leiterman, *Comment: Smart Companies, Foolish Choices? Product Designs that Harm Competitors,* 15 SANTA CLARA COMPUTER & HIGH TECH. L.J. 159 (1999) (collecting and examining related cases).

[21] The others are the "Finished Product Rationale," ¶1748, the "Intellectual Property and Other Government-Encouraged Bundling," ¶1749, and the "Phantom Separate Product" rationale, ¶1750.

1. Competitive Market Practices [CMP]

The core of the Areeda analysis is the "Competitive Market Practices" test. AREEDA TREATISE ¶1744. The aim of this test is to determine whether competitive firms, operating in a competitive environment, bundle products in the way that the defendant does. The answer to this question provides good indirect evidence of whether it is efficient to provide the two items as two products. If firms with no market power, operating in a competitive environment, similarly bundle the two items, then a court has good reason to believe that these two items cannot be efficiently offered separately. If they could, then the competitive market would provide them separately. The actual market practices of firms without market power is therefore a good indicator of whether two items can be offered separately efficiently.

To apply the CMP test, however, a court must first determine what the relevant "bundle" is. While it is true that other competitors "bundled" browsers with their operating system, for purposes of the CMP test, I don't believe that practice should count. Including free and easily removable software with an operating system or with any software package is not the kind of "tying behavior" at issue in a Section 1 case. So long as the consumer, or OEM, can easily remove the allegedly "tied" product, there can be no "forcing" for purposes of *Jefferson Parish*.

The only relevant alleged tying behavior would be a case where a manufacturer bundled a browser which the consumer or manufacturer could not easily remove. So conceived, as this Court has found, ¶153, no operating system manufacturer except Microsoft bundles products in this way. Only Microsoft, this Court has found, hampers that choice. It interferes with that choice both by overriding a default selected by the user, ¶171, and by forbidding OEMs from removing the obvious access to the IE technology. ¶158. There is therefore no competitive market practice that would suggest that Microsoft's two products should be considered a "single product" for purposes of antitrust tying law.[22]

It might seem odd to conceive of a tie as constituted by the failure to permit the removal of software product. This oddness, however, is a function of the nature of software. When a computer is sold, its

hard disk is an underutilized shipping container. To the extent it is empty, it represents software that could have been offered to consumers. Many software producers distribute software on this "container," requiring the purchaser to register for a key after a short time, or otherwise charging for the software later. The relevant forcing in a market such as this is the refusal to allow the consumer the option to decline the offer.

2. New Product Rationale

The second relevant test for determining whether this Court should consider the two items a "single product" for purposes of antitrust tying law is the "New Product Rationale." See AREEDA TREATISE, at ¶1746. This test is used to determine whether two products, bundled together in a new way, ought to be treated as a "single product." If the test is satisfied, then two items will be considered "one product" and not subject to tying scrutiny.

While the origin of the "New Product Rationale" is *United States* v. *Jerrold Electronics Corp.*, relied upon by the Supreme Court in *Jefferson Parish*, the bulk of the authority relied upon by the Areeda authors are cases decided prior to *Jefferson Parish* and, in particular, prior to the Supreme Court's rejection of the "functional integration" test for one or two products.[23] No case since *Jefferson Parish* has relied upon this rationale — except the Court of Appeals in *Microsoft II*. Except for *Microsoft II*, only one case relied upon by the authors involved software ties,[24] and the one case referring to the test since *Jefferson Parish*

---

[22] The evidence about the BeOS is not to the contrary. While the Court has not made direct findings on this issue, the evidence appears to be that the BeOS simply requires a HTML reader to render its help files; it does not require any particular reader. Pl. Findings ¶116.2.5.2.

[23] The treatise cites *Telex Corp.* v. *IBM,* 367 F. Supp. 258, 347 (N.D. Okla. 1973); *ILC Peripherals Leasing Corp.* v. *IBM Corp.*, 448 F. Supp. 228, 230-32 (N.D. Cal. 1978) (disk drive unit and head/disk assembly); *Innovation Data Processing* v. *IBM Corp.,* 585 F. Supp. 1470, 1476 (D. N.J. 1984) (software); *Information Resources, Inc.,* v. *A.C. Nielsen Co.,* 615 F. Supp. 125, 129-30 (N.D. I. 1984) (Optical scanning data integrated with manual data), finding one product, and *Anderson Foreign Motors* v. *New England Toyota Distributors,* 475 F. Supp. 973, 983 (D. Mass. 1979), *Data General Corp. Antitrust Litigation*, 490 F. Supp. 1089, 1108-09 (N.D. Cal. 1980), and *Multistate Legal Studies* v. *Harcourt Brace*, 63 F.3d 1540, 1551 n.10 (10th Cir. 1995), finding two products. The treatise cites *Montgomery County Assoc. of Realtors.* v. *Realty Photo Master Corp.*, 783 F. Supp. 952 (D. Md. 1992), which did find a single product for a software product that tied pictures to descriptions, but the Court reached this conclusion under the "separate demand" test of *Jefferson Parish.* The IBM peripheral cases coming from the 9th Circuit were influenced by that circuit's "function of the aggregation" test for determining whether there was one product or two. See *ILC*, *supra*, at 230. But the Supreme Court in *Jefferson Parish* expressly overruled this "function" test.

[24] See *Innovation Data Processing*, *supra*.

draws its continued validity into some doubt.[25] Thus one might well question whether the rule survives *Jefferson Parish* at all, or at least question the weight the rationale gives to the "new product" conclusion in light of the Supreme Court's rejection of the "functional integration" test.

Nonetheless, the motivation for the rule does make sense, especially in the context of software products. As Microsoft rightly argues, Df. CL., at 5, MS Findings, at ¶529, innovation in software often involves the bundling of functionality into a product that was not included in the product before. If the defendant is the first to bundle this new functionality, then there would be no historical practice against which to compare the bundle. And if there were no historical practice, then the CMP test would no longer be a useful proxy for determining whether it was efficient to provide the two items separately.

How this test should be applied in the context of software products, however, is uncertain — indeed, how this test should be applied in the context of the software products at issue in this case in particular is uncertain. While the Court of Appeals, using the rationale, concluded the text meant that Windows 95 and IE were likely "integrated," the 1998 edition of the treatise, applying the same test, agreed with this Court that they should likely be considered "separate products," HERBERT H. HOVENKAMP, 1998 SUPPLEMENT TO AREEDA, ET AL., ANTITRUST LAW, ¶1746b, at 467-69 (1998), and the 1999 edition of the treatise, applying the same test again, agreed with the Court of Appeals, though for different reasons, that they should be considered "integrated." 1999 SUPPLEMENT, ¶1746.1b, at 494.

These different conclusions reflect a genuine uncertainty about how to apply the "New Product Rationale," and the cases underlying it, to software products.[26] And combined with the considerations that I began with above — both the observation that the opportunity for strategic bundling is greater with software than with "technology" and the objective that the test not bias incentives in the design process — they raise a difficult question about whether the test should survive at all. But rather than reject the test completely, the experience of this Court suggests a relatively simple modification to the "New Product Rationale" that would adjust it to the "unique problem" that "computer software poses." 1999

[25] See *Multistate*, 63 F.3d, at 1551 n.10.

SUPPLEMENT, ¶1746.1b, at 491. To see this, however, will require a more extensive explication of the Areeda analysis.

The "New Product Rationale" is two tests bundled into one. As the Areeda Treatise puts it, courts should treat two products as one for purposes of antitrust tying law if:

> (1) no relevant analogue of bundling or unbundling exists because the defendant's bundle causes the items to operate together in a way that had not been tried before *or* (2) the items were previously sold unbundled and operated together but the newly bundled items operate better when bundled *by the defendant* than if bundled by the end user.

AREEDA TREATISE, ¶1746, at 224. The Court of Appeals applied the second disjunct to interpret the term "integrated" for purposes of the consent decree. 147 F.3d, at 949. But the motivation for the test is not whether two products are, in the abstract, "integrated."[27] Indeed, *Jefferson Parish* expressly rejects the notion that "functional integration" affects the "separate product" test.[28] Instead, the core of ¶1746 turns on whether there is a special reason why, though products are presumptively "separate" under ¶1743, the defendant should, nonetheless, be permitted to bundle them.

---

[26] Or at least a conflict among the authors. The 1998 and 1999 Supplements are authored by Professor Hovenkamp.

[27] The term "integrated" had a life of its own at the trial. Some spoke of "deep integration," "deeper integration," "light integration," "close integration," "tight integration," "technical integration," "seamless form of integration," "technical seamless integration" and "the tight welding of code." One might order the different senses of integration conceptually as follows: At one extreme, a program could be said to be "integrated" with an operating system if it would run on that operating system. Professor Fisher referred to this sense to reject it as a true sense of "integration." January 12, 1999, P.M. Session, 1999 WL 11492, at *1. A stronger sense of "integration" describes two programs that allow seamless transfer of data between them — for example, Microsoft Word and Excel, "work well together," as Professor Felton described them. June 10, 1999, A.M. Session, 1999 WL 380890, at *11. A stronger sense again would be two programs that shared code. This is the phase two integration that I described in the relevant facts, and it is what Professor Schmalensee referred to as tighter integration, where the degree of "tightness" was a function of the amount of shared code. January 19, 1999, P.M. Session, 1999 WL 20651, at *11. Microsoft's Allchin referred to this sense of integration as well, focusing on the efficiency that results from "reduc[ing] the amount of code." February 2, 1999, P.M. Session, 1999 WL 47198, at *9. And finally, there was "integration" in the sense of programs written to share code, and which could not be separated without disabling the other. This is phase three integration, which Professor Fisher called the "tight welding of code." January 7, 1999, A.M. Session, 1999 WL 6529, at *9.

[28] This understanding was confirmed by the 10th Circuit in *Multistate, supra*, 63 F.3d, at 1547. Microsoft distinguishes this express rejection of the "integration" test by saying the Court was not considering "integrated products" but instead was considering a "functionally integrated package of services." Df. CL., at 4. But the "products" at stake in *Jefferson Parish* were "services"; the alleged tie was between the service of surgery and the service of anesthesiology. So unless there is some principled reason not to consider "integration" in the context of alleged ties of services, but to consider integration in the context of alleged ties of software products, the Court's rejection of an "integration" defense would seem dispositive.

Two types of "special reason" are accounted for by the treatise. In the second, the issue is whether it is *necessary* for the defendant to bundle the two items for the functionality of the two items operating together to be realized. The answer to this question turns not on how well the bundle functions relative to other bundles,[29] but on *whether it is necessary for the defendant to bundle the products for whatever efficiency there is to be achieved*. AREEDA TREATISE, ¶1746, at 227. As the section describes with a hypothetical that matches some of the facts this Court has found:

> [C]onsider 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.

AREEDA TREATISE, ¶1746b, at 229 (emphasis added). As the government argued, Pl. Findings, at ¶¶159.4.1, 159.4.2, however, and as this Court has found, ¶190, applying that test in the context of the this case would entail that Windows 98, for example, was not a "new product," since essentially the same functionality could have been achieved by the consumer whether the consumer purchased Windows 98 from Microsoft, or purchased Windows 95 and installed IE 5.0. No difference in the functionality of IE, in other words, would have existed had Microsoft allowed consumers the choice. Thus, at least for some versions of IE, the "New Product Rationale" would provide no reason to find that the two items were a "single product" for purposes of antitrust tying law.

The latest edition of the Areeda Treatise, however, is skeptical about the application of that standard to software products. 1999 SUPPLEMENT, ¶1746.1, at 487. Its skepticism derives again from the nature of software. Animating this second disjunct of the rationale is the view that "generally, two items will operate better when bundled by the defendant because the bundling requires some physical or technological interlinkage that the customer cannot perform." AREEDA TREATISE, ¶1746b, at 227. This may

[29]Not relative to a competitor's product, as the Court of Appeals indicated in *Microsoft II*, assuming again that opinion is to be read to interpret antitrust tying law. 147 F.3d, at 950.

well be true for physical products,[30] and certainly for many intangible products. But, subject to a qualification that I raise below, it not true in the same sense with software products. With software products, whether two items will operate better when bundled by the defendant than when bundled by the customer is simply a matter of software design. If there is a difference in the functionality of the resulting bundle if installed by the consumer, this is because the designer chose to make it that way. 1999 SUPPLEMENT, at 493-94. And as God can't plead the laws of nature as a defense, it would be odd to build an immunity based on an impossibility that the defendant itself created.

Physical or technological products are different. 1999 SUPPLEMENT, ¶1746.1b, at 493. If a car manufacturer bundles a bumper with the car, or an automatic transmission with the car, this is a bundle better bundled by the manufacturer because bumpers and transmissions are heavy and large objects to handle — not because the manufacturer has made them that way, but because nature has made them that way. Likewise, installing them properly requires training. There is no "code" that could provide the training separately. Thus there is no similar ability of the defendant with non-software products to as easily architect the bundle so as to satisfy the rule of ¶1746b. ("One could not so readily design an automobile whose pistons could be installed by the end user as easily as by the factory assembler.") 1999 SUPPLEMENT, ¶1746.1, at 488-89.

One response to this feature of software would be to interpret this part of the "New Product Rationale" to ask not whether two products *can be* combined by the consumer as successfully as by the defendant, but whether they *could be* designed to be combined by the consumer as successfully as by the defendant. This seems to be the meaning of ¶1743 of the Areeda Treatise, but the latest version of the treatise rejects that interpretation. Citing the plasticity of software, the author of the 1999 Supplement concludes "[t]he problem with such a test is that … incremental software would *never* qualify as part of the same product but must be considered separate." 1999 SUPPLEMENT, ¶1746.1b, at 494. "[V]irtually all improvements to software would have to be regarded as 'separate products.'" *Id.*, at 489.

[30] It explains, for example, the IBM peripheral litigation cases. See cases cited *supra*, note 21.

The latter conclusion does not necessarily follow on the Areeda analysis,[31] but even if it does, there is a different problem raised by interpreting the test to ask not whether two products as designed *can be* combined as successfully by the consumer as by the defendant, but rather whether they *could* be designed to be. For while software is plastic, as the evidence in this case suggests, it is too simple to assume (as the treatise does) that any combination is possible. It might be that an update to a program is so significant, and the range of different installations of a program so vast, that it would be more efficient to perform the update at the factory, so to speak, rather than permitting the consumer the choice. One can't know in the abstract. So to evaluate whether this is true, a court would have to investigate the character of the two products — to examine, as this Court has, the nature of the interaction between the two products, and the character of the process that combines, or might combine, the two products. But to require courts to engage in that inquiry would defeat the rule as a "screening" device. To apply ¶1746b would require an extensive factual inquiry into what "could be" done, contrary to the purpose of ¶1746b and consideration (3) above that the inquiry be relatively simple.

These considerations suggest that at least the second disjunct of the "New Product Rationale" is not usefully applied in the context of software products. To the extent that it turns on the technology of bundling, it either requires a judicial review that is too extensive, or it creates an incentive for designers to architect their software to fit an antitrust rule. Both results counsel against embracing this "special reason" for treating two products as a "single product."

---

[31] The reason is that two products, though "separate products" for purposes of the "New Product Rationale," could still be "single products" for purposes of the "Competitive Market Practices" test, or any of the other rationales. So for example, the 1999 edition of the Areeda Treatise suggests that if the two software products were considered "two products" because they could be offered separately and then combined, this would mean that Windows 95 itself was not a "single product" but was instead an impermissible tie of MS-DOS and Windows 3.x. 1999 SUPPLEMENT, ¶1746.1, at 489. The same claim was recently made against Microsoft in the *Caldera* litigation. *Caldera, supra*. In that case, because there was a dispute about whether in fact there was technological integration between the then current versions of Windows and MS-DOS, the District Court permitted the issue to go to trial. According to deposition testimony, and contrary to the assumption of the Court of Appeals in *Microsoft II*, in fact there was no "shared code" between the two products. But in both that case, and under the consent decree case, this analysis misses the first step of the Areeda test — whether it was a competitive market practice to bundle a GUI interface with the operating system. Under that test, it would seem the dominant and emerging practice of other OS firms was to bundle the GUI interface (Macintosh, and OS/2), and that the same bundle by Microsoft should be considered a "single product" — again, whether or not technologically the package shared code or was "truly integrated."

The second "special reason" outlined by ¶1746 of the Areeda Treatise for treating two items as a "single product" under the "New Product Rationale" (¶1746a) does not turn upon how the two products are or could be bundled. Here the question is whether "no relevant analogue of bundling or unbundling exists because the defendant's bundle causes the items to operate together in a way that had not been tried before." AREEDA TREATISE, ¶1746a, at 224. The case animating this test is *United States* v. *Jerrold Electronics*, 187 F. Supp. 545 (E.D. Pa 1960), relied upon by the Supreme Court in *Jefferson Parish*, where Judge Van Dusen concluded that in a nascent market, the alleged bundling was justified, but at an unspecified point later in the course of the case, it was no longer justified. *Id.* at 560. The aim of the test is to protect this innovative new design, at least so long as it is new.[32]

The applicability of this test to the facts of this case might be questioned, because obviously, browsers and operating systems have been functioning together for almost a decade. But as Microsoft claims, Microsoft has bundled its browser technology in a way that has not been tried before — for browsers, at least. Specifically, by choosing to componentize its browser functionality, and thereby expose a larger range of APIs to other applications, Microsoft has caused its browser product to operate with its operating system product in a "new way." Thus to the extent there is a relevant "New Product Rationale" for software products, it would appear that the analysis of ¶1746a is the better aspect of that rationale to consider Windows 95/98 under.

But in the context of software products, this disjunct suffers from a flaw that is analogous to the flaw with ¶1746b as applied to software. For again, because of the character of software, it is always possible to bundle software such that it operates in a "new way." AREEDA TREATISE, ¶1746a; 1999 SUPPLEMENT, at 494. This consideration led the Court of Appeals to look to other factors that might limit the scope of its interpretation of "integrated products" — ways, that is, of distinguishing integrated products that would reflect genuine improvements from new products that were merely pretextual. The

[32] Microsoft argues that "[n]o court has ever sustained such a challenge to a single integrated product." Df. CL., at 2. That is not completely correct. In *Jerrold Electronics*, while the court held that a television transmission system was properly considered a "single-product" during its initial period of operation, it did sustain a challenge of that same "single-product" after the introductory period passed. 187 F. Supp., at 560.

exclusion of mere "bolting" and the embrace of designs that bring "some good" are two ways to filter new product designs to assure that the test does not immunize bundles too broadly.

But these limitations just lead the court down a path that has no obvious stopping point. Critics of the Court of Appeals' standard believe it is not extensive enough to capture strategic bundling behavior. Other Courts have therefore suggested a more extensive inquiry into the character of the "integration." *Caldera, supra,* at 1319. And to the extent that the inquiry becomes more invasive, the less the test looks like a "screening" test, and the more it looks like a full blown analysis into the merits of the design.

As I described above, these considerations have led the author of the most recent supplement to the Areeda Treatise to recommend the finding of a "single product" "in all cases in which the code for the two programs is interspersed such that the purchaser cannot readily separate them." 1999 SUPPLEMENT, ¶1746.1b, at 494. As I said, if the Court embraced this view, then the Court would conclude that Windows 95/98 is a "single product" for purposes of Section 1 tying, and Microsoft would prevail.

In my view, however, the evidence as this Court has found it suggests a more straightforward approach to the question of new software products — one that does not lose "most of the advantages of integration" or create the incentive for software producers to "comply with" the rule "by interspersing code." *Id.* But one that does account for the special factors about software that Microsoft advances.

Under this approach, two software products combined in a "new way" would be considered a "single product" for purposes of antitrust tying law — regardless of how they were linked (whether by code or by contract) — but this conclusion would be presumptive only. The presumption could be defeated if the plaintiff can show that the particular bundle at stake raises the risk of a particular anticompetitive harm. If, in other words, it is a bundle of the kind of products likely to cause an anticompetitive harm, then the presumption finding a "single product" would be rebutted.

What the risks that would defeat this presumption are I will describe in the next section. But I want first to highlight the reasons why a test different from the current version of the Areeda test seems especially appropriate in light of the facts that this Court has found.

As the treatise acknowledges, the effect of the "New Product Rationale" will be to create an incentive for software producers to modify their code to fit within the terms of the rule. *Id.* This, in my view, is a significant weakness of the rule. So long as there is a risk of strategic bundling, the law should not create an incentive for the bundler to hide that bundle in code. As this Court has found, such pretextual bundling is both inefficient and costly. ¶173, 174. The law has no reason to create the incentive for that inefficiency.

Instead, the "New Product Rationale" should be neutral between "new combinations" that are effected by contract or effected by code. If there are two software products that could be combined to operate together "in a new way," then so long as there is no risk of strategic bundling, the law should allow the innovator to decide how the two products are more efficiently combined. The aim of any antitrust inquiry should be whether the particular bundle is a strategic bundle, aiming at anticompetitive ends, not whether the bundle achieves it interlinkage through contract or software.

The better solution is to keep the court's focus above the hood, so to speak, and probe the risk of anticompetitive bundling of software through different techniques. That is the aim of the modification to the "New Product Rationale" that I believe this Court could adopt. Under this modification, software manufacturers who show that they have bundled two software products together in a "new way" should get the benefit of this "New Product Rationale" whether their bundle achieves this benefit through "technological integration" or not. But the benefit of this rule — treating two products as one — should be presumptive only, subject to being defeated by the other considerations that I describe below.

The consequence of this rule would be that software bundling generally would constitute a single product, so long as the defendant can establish that the bundling operates together in a "new way." This would mean that in the general case, a software producer would be free to add functionality to its software products, whether or not that inclusion meant other software producers would be disadvantaged. As the history of this case shows, there are many examples of improvements to the operating system that have eliminated the demand for other products in the software industry.

But this Court has implicitly accepted this consequence by dismissing the States' Section 2 claim of "monopoly leveraging" against Microsoft. *United States* v. *Microsoft Corp.*, 1998 WL 614485 (D.D.C. 1998), at *27. As the Court said there, in the ordinary case, antitrust law has little reason to fear this form of bundling, even if the consequence is that competitors are driven out of business. The general rule should be a lenient one, to encourage innovation in the design of software products; the exception should be where there is good reason to fear that the bundle will create anticompetitive harm.

Under this approach to the "New Product Rationale," then, Microsoft's operating system and browser functionality should be treated presumptively as a "single product" for purposes of antitrust tying law, unless an independent reason exists why this type of bundle raises special anticompetitive concerns. I consider one such concern in the next section.

3. Same Product Rationale

The third rationale relevant to this case is the "Same Product Rationale." AREEDA TREATISE, at ¶1747. Like the "New Product Rationale," this test involves two inquiries, though only one directly supports the rationale. The core of the rationale recommends that if the defendant is bundling "complete substitutes," then the two products should be considered "one product" for purposes of tying law. But the rationale is quick to make clear that if the defendant is bundling "partial substitutes," then the "same product rationale" does not apply. If the products are partial substitutes, then they should be considered "two products" for purposes of tying law. AREEDA TREATISE, at 232.

The reason for the difference lies in the differences in competitive harm likely to arise from the bundle. "[A] tie of partial substitutes might leverage market power and spread inefficiency from one 'market' to another. [O]nly in the partial substitute case can tying *spread* the *inefficiency* to the new market." AREEDA TREATISE, ¶1747, at 232. Thus, while complete substitutes bundled together are typically not harmful to the competitive process, "bundles of partial substitutes are *more* dangerous than the typical tie." AREEDA TREATISE, at ¶1747. Bundles of partial substitutes, Professor Hovenkamp notes, can "sabotage a nascent technology that might compete with the tying product but for its foreclosure from the market." 1999 SUPPLEMENT, ¶1746.1d, at 495.

This insight into the dangers of "partial substitutes" provides a check on the scope of the "New Product Rationale." Based on the facts that this Court has found, in light of the Supreme Court's injunction that the "single product" inquiry must be "based on whether there is a possibility that the economic effect of the arrangement is that condemned by the rule against tying," *Jefferson Parish*, 466 U.S., at 21, this Court could condition the "New Product Rationale" as applied to software products upon whether the product was a "partial substitute" for the tying product. If it is a "partial substitute" that the plaintiff can show will likely have an anticompetitive impact, then the bundle should not be considered a "single product" for purposes of antitrust tying.

This rationale has a direct application to the Court's findings in this case. The theory of the government's case, and the core of the findings that this Court has made, is that Microsoft exercised its power to protect what has been termed the "applications barrier to entry." ¶31, 36-44. By maintaining control over the APIs that independent software vendors write to, Microsoft is in a position to maintain the dependency the market has on its Windows operating system. ¶68.

One aspect of Microsoft's campaign, this Court has found, has been to assure that the browser technology that Windows users deploy is a browser technology that it controls. ¶227. In this way, it can assure that the browser doesn't become a competitor to the Windows API set, or at least that, to the extent the browser exposes APIs, they are APIs that Microsoft controls as well. ¶79. By bundling a browser that itself is a partial substitute for the Windows platform, but which Microsoft nonetheless controls, Microsoft is able, this Court has found, to avoid losing developers to a foreign API set.

In this way, "browser technology" has the potential to be a partial substitute for the Windows operating system. It is not itself a complete substitute — as this Court has found, browsers do not expose anywhere close to the number of APIs that an operating system does. ¶77. But to the extent that they expose any significant set of APIs, sufficient to induce developers to write applications to the browser APIs, they threaten to reduce the market power of the underlying tying product — Windows.

Applying the rationale of the partial substitutes test to the facts of this case, the Court could therefore conclude that the presumption of the New Product Rationale — that the browser product and

operating system should be considered a "single product" — has been rebutted, and the two software products of an operating system and browser functionality should be considered as "two products" for purposes of antitrust tying.

While application of this test, given the current design of Microsoft's operating system, should entail a finding of "separate products," as a general matter software developers should be able to avoid the risk of this "partial substitute" rationale simply by designing their software so as to give consumers an option not to take the partial substitute. Like the free but optional browsers discussed in the CMP test above, if the product were easily removable there would be no antitrust forcing. This was apparently IBM's strategy with software bundling, see *Innovation Data Processing* v. *IBM Corp.*, 585 F. Supp. 1470, 1476 (D. N.J. 1984) (two software products bundled together not considered tied where customer had the option to remove one, and cancel the license for that component), though the court in that case did, in dicta, suggest that that the removability did not matter to the test. Again, however, this case arose before *Jefferson Parish*.[33]

Finally, nothing would require that the presumption in the "New Product Rationale" as applied to software could be rebutted only in the case of partial substitutes. Obviously, there may be other objective and significant competitive threats that merit similar treatment.[34] But this Court need not write a Treatise. It is enough to apply a reasonable rule in light of the facts that this Court has found.

D. A SUMMARY OF THE TEST APPLIED

If *Microsoft II* does not control, then the Court should decide the Section 1 tying claim under the standards of *Jefferson Parish*. While case law before *Jefferson Parish* suggested that the "separate product" question may be affected by "integration," *Jefferson Parish* makes the "separate product" analysis turn upon

[33] The Areeda Treatise points out the pricing problem with a solution that permits a finding of a "single product" if the defendant permits opt-out. See AREEDA TREATISE, ¶1743b, at 194-95. Since the product at issue here is zero-priced, this Court need not resolve that question in this case.

[34] For example, if the tied product raises rivals' costs sufficiently to make competition in the tying market impossible, then again a court may have reason to find the presumption that these two products are one rebutted. See, e.g., Pl. CL., at 64; AREEDA TREATISE, ¶1729b; Thomas Krattenmaker & Steven Salop, *Anticompetitive Exclusion: Raising Rivals' Costs to Achieve Power Over Price,* 96 YALE L.J. 209, 230-48 (1986); Meese, *Bundling*, *supra* note 13, at 108-10.

"separate demand." Applying the "separate demand" test to the facts as this Court has found them, the Court should conclude that the government has established "separate products" for purposes of Section 1 tying.

It is also appropriate, however, for the Court to suggest in addition a test to apply to software ties that reflects the experience of this case. Based on the framework of the Areeda Treatise, I have suggested an analysis that reflects the findings this Court has made, and the rightful concerns about innovation that Microsoft has raised. I have argued:

> (1) A "Software Product" should be identified as "functionality separately valued by consumers." This Court has therefore found that "browser functionality" can be considered a "software product."
>
> (2) The "Competitive Market Practices" test indicates that while it is not improper to bundle an optional browser with an operating system, there is no precedent among competitors to validate tilting the operating system towards a particular browser. Thus, the CMP test would provide no reason to treat this kind of "browser functionality" combined with operating systems as a single product.
>
> (3) Under a modified "New Product Rationale," however, there would be a reason to treat the browser functionality and operating system presumptively as a single product.
>
> (4) But given the competitive threat posed by the bundling of "partial substitutes," that presumption would be rebutted, as the alleged tied product, browser functionality, operates in this market as a partial substitute for an operating system.

"Browser technology" and "operating system" would therefore be considered two products for purposes of antitrust tying.[35]

---

[35] I have not addressed whether this modified "New Product Rationale" would entail a "rule of reason" analysis or a modification of the "per se" rule, but I do not believe the test entails a "rule of reason" analysis. While the "separate product" inquiry may be more extensive under a rule of reason, AREEDA TREATISE, ¶¶1742b, 1742c, and hence more in line with the modified "New Product Rationale," the aim of the modified rule is to narrow the over-inclusiveness of the rule, and may therefore be an appropriate balance in the context of the per se rule as well. In any case, the Supreme Court has indicated that the ultimate aim under both standards is the same. See *NCAA* v. *Bd. of Regents, University of Oklahoma*, 468 U.S. 85, 103-04 n.26 (1984).

CONCLUSION

For the foregoing reasons, in the manner described, this Court may conclude either that the government has succeeded in establishing a "separate product" for purposes of "tying" under Section 1 of the Sherman Act, or that it has not.

Respectfully submitted,

Lawrence Lessig, *pro hac vice*

Harvard Law School
1525 Massachusetts Ave
Cambridge, MA 02138
(617) 495-8399

February 1, 2000

Amicus Curiae

CERTIFICATE OF SERVICE

I hereby certify that on this 1st day of February, 2000, I caused a true and correct copy of the foregoing Brief of Amicus Curiae to be mailed by U.S. First Class Mail to:

A. Douglas Melamed, Esq.
Antitrust Division
U.S. Department of Justice
950 Pennsylvania Avenue, N.W.
Washington, D.C. 20530

John L. Warden
Sullivan & Cromwell
125 Broad Street
New York, NY 10004
(212)-558-4000

Richard L. Schwartz, Esq.
Deputy Chief, Antitrust Bureau
New York State Attorney General's Office
120 Broadway, Suite 2601
New York, New York 10271

Kevin J. O'Connor, Esq.
Office of the Attorney General of Wisconsin
P.O. Box 7857
123 West Washington Avenue
Madison, Wisconsin 53703-7957
Fax: (608) 267-2223

Christine Rosso, Esq.
Chief, Antitrust Bureau
Illinois Attorney General's Office
100 West Randolph Street, 13th Floor
Chicago, Illinois 60601
Fax: (312) 814-2549

_____
Lawrence Lessig