Scheduled for Oral Argument on February 26 and 27, 2001

In The
United States Court of Appeals
For The District Of Columbia Circuit

No. 00-5212 (Consolidated with No. 00-5213)

United States of America,
Plaintiff-Appellee,
v.
Microsoft Corporation,
Defendant-Appellant.

On Appeal from the United States District Court
for the District of Columbia

Brief of Professor Lee A. Hollaar
as *Amicus Curiae*
in Support of Neither Side

-

Professor Lee A. Hollaar, *pro se*
School of Computing
University of Utah
3190 Merrill Engineering Building
50 S Central Campus Dr
Salt Lake City UT 84112-9205
Phone: 801-581-3203
801-363-8086
Fax: 801-581-5843

December 26, 2000

Certificate as to Parties, Rulings, and Related Cases

Parties and *Amici*

All parties, intervenors, and *amici* appearing before the District Court and this Court are listed in the Brief for Microsoft Corporation.

Rulings Under Review

References to the rulings at issue appear in the Brief for Microsoft Corporation.

Related Cases

References to related cases appear in the Brief for Microsoft Corporation

Table of Contents

Table of Authorities

Cases

Statutes

Glossary

"API"	An "application programming interface" is a specified way for an application program to make use of the services provided by an operating system. These may be calls directly invoking the basic functionality of the operating system, or calls to shared library routines distributed with the operating system that perform more complex operations than the direct calls to the operating system, such as rendering a page of text.
"DLL"	A "dynamic-link library" is Microsoft's term for a shared library where many applications can use the same subprogram in the library and that subprogram does not need to be in memory unless it is actually being used. A DLL behaves the same as if it were a subroutine of the application program, making calls to the operating system when necessary.
"HTML"	"Hypertext Markup Language" is the way that documents are formatted for use on the Internet's World Wide Web.
"OEM"	An "original equipment manufacturer" is the maker of computer systems for sale to the public. They generally take hardware components designed and manufactured by them and others, and install an operating system and application programs to get a system that the public can use with little additional effort.
"Windows"	When used without qualification, I am referring to the Windows 95 operating system from Microsoft or its later versions, Windows 98 and Windows ME.

Introduction and Interest of *Amicus*

In this brief, I will be discussing the combining of two (or more) computer programs to form a single program which replaces the prior programs in the market. (I will be using the neutral term “combining,” rather than “tying,” ”bundling,” “integrating,” or similar terms that have implications that may prejudge the nature of the combination.) I will not be discussing such combining from an antitrust law point of view – I’m sure that will be well-covered by the other briefs – but in light of the applicable computer software technology.

As a professor of computer science teaching hardware and software design for about twenty-five years and law-and-technology for over fifteen years, and as the primary technical expert for the plaintiff in *Caldera v. Microsoft* (United States District Court, District of Utah, Central Division, Civil No. 2:96CV645B), I have given substantial thought to the nature of computer software and the technological effects of combining two programs.

Other than some Microsoft stock held by mutual funds over which I have no control, I have no financial interest in the outcome of this appeal, and support neither party in this brief. And while I have been a technical expert in cases against Microsoft, in the *Caldera* case Microsoft first tried to hire me and I made my decision to work for Caldera after presentations of the merits of their positions by both Caldera and Microsoft.

While I have identified myself as a Professor of Computer Science in the School of Computing at the University of Utah, I am filing this brief as an

individual, and not as a representative of the University of Utah. The views expressed in this brief are solely my own, and not those of any other person or organization, and no other person or organization has provided financial support to me for this filing.

I am submitting this brief because I believe that this Court's opinion must be both legally and technically sound. As an example from a related area – copyright law – the Third Circuit's opinion in *Whelan v. Jaslow*, 797 F,2d 1222 (1986), was based on an unrealistically simple view of computer software: that a program has one (or a very few) ideas, and everything else is protectable expression. The opinion was heavily-criticized by both technical and legal commentators. In contrast, the Second Circuit's opinion in *Computer Associates v. Altai*, 982 F.2d 693 (1992), is based on a better computer science foundation. Although the tests from that opinion are not as simple as those in *Whelan*, their complexity accurately mirrors the complexity of the computer software, and so have been well-received by both courts and commentators.

Operating Systems

Before discussing the combining of computer programs, it will be useful to look at the special nature of operating systems such as Windows, that sets them apart from other computer programs when we consider the combining of programs. (When I use the term Windows without any further designation, I'm referring to either Windows 95 or its later versions, Windows 98 and Windows ME.)

First, I will discuss how there is no good definition for what is included in an operating system, and that in a broad sense an operating system might include shared libraries and application programs in addition to the program that controls the computer. Then, I discuss how the services requested by application programs can be supplied either by the basic functionality of the operating system, or through library routines shipped with the operating system. Such library routines are how Microsoft has included its Internet Explorer capability in Windows 98.

As Microsoft says on page 13 of their brief, “There is no universally accepted definition of operating systems.” Clearly, it is not useful to base legal decisions on terms which have a variety of meanings, such as “operating system,” “integrated,” or “Windows.” You have to see what is actually there.

For example, “Windows” can refer to anything from a 32-bit operating system that also includes a version of MS-DOS for operating system loading and to provide support for legacy programs, to everything included on the distribution disk that contains that operating system. Because there is no clear definition of what an operating system is, it is difficult to say whether an application program shipped with an operating system is part of that operating system or not.

As another example where ill-defined terminology makes it impossible to determine what is included, when Microsoft refers to its intention to “include built-in access to the Internet” on page 21 of their brief, it is not clear whether they are referring to their Internet Explorer browser program (not included in the original release of Windows 95, see page 24 of Microsoft’s brief) or the other networking

capabilities that they included in the first release of Windows 95: support for the Internet data communications protocols, support for dial-up connections to an internet service provider, and utility programs for remote access of other computers such as FTP (file transfer program) and Telnet.

Contemporary operating systems are not a single program, and provide services to application programs in a variety of ways. Sometimes the application program uses the basic functionality of the operating system, as when it opens a file and reads data from it. But other times it simply calls a library routine that was included with the operating system, to perform a more complex operation such as formatting a windows to display all the files in a particular directory so that a file can be selected for opening.

These library routines behave the same as if they were subroutine components of the application program, making calls to the operating system when necessary. However, these libraries are managed in such a way that they can be shared between a number of programs, reducing system memory requirements, and are linked to the application program whenever they are needed, rather than always being a part of the application program. Because these library routines are linked dynamically (rather than statically), Microsoft refers to them as dynamic-link libraries, or DLLs. Other operating system suppliers call them shared libraries.

It is precisely such DLLs that Microsoft refers to when it talks about the “componentized design” of Internet Explorer starting with version 3 (see page 31 of Microsoft’s brief), and the ability for other application program developers to call

those DLLs the same way Microsoft does in Internet Explorer. (Microsoft indicates that the DLLs they have provided are MSHTML.DLL, SHDOCVW.DLL, URLMON.DLL, and WININET.DLL. See the descriptions of these shared libraries in the Glossary and on page 31 of Microsoft's brief.)

Microsoft is not the only supplier of DLLs for Windows. Many application programs, including Netscape and WordPerfect, have their own DLLs to supply functionality when needed while not requiring memory when the functionality is not being used. These can be readily seen by looking in the directory where, for example, Netscape has been installed. Other vendors supply DLLs to application program developers for performing particular functions, so that those developers do not have to reinvent the wheel to use those functions. Microsoft is not in a unique position to supply an HTML rendering engine as a DLL, except to the extent that it can put it on the same distribution disk as Windows.

In addition to the DLLs for Internet Explorer and a host of other functions, Microsoft's distribution of Windows contains a variety of other things that are not a necessary part of the operating system. About 30 different application programs or features can be selected for inclusion or exclusion at the time Windows is installed. These include the infamous solitaire game that acts as a time sink for many people, a simple word processor, dial-up networking, and an email client. Since these programs can be excluded at the time Windows is installed, and can be removed at any later time, without affecting the operation of Windows (other than the program no long being available), they are clearly an adjunct to the Windows operating

system although they are included as part of the Windows distribution. Many are simply application programs distributed with Windows.

Until Windows 98, at least those portions of Internet Explorer seen by the user as a browser application program were removable from Windows in the same fashion as the solitaire game and similar application programs. Removal of any application program, whether from Microsoft or other vendors, does not remove all the DLLs used by that application program when there is another application program that is using a DLL.

The relationship between an operating system and a program that uses the service of that operating system is fundamentally different from the relationship between two application programs. An application program written for a particular operating system always needs that operating system in order to run. (It could also run on a compatible operating system that has the same application program interfaces (APIs), but such a system is virtually impossible to create today because of the thousands of APIs provided by an operating system like Windows, with more added every year.) That is generally not the case for two application programs: in the normal course of using the Microsoft Excel spreadsheet program, you don't need Microsoft Word installed on your machine, but you will not be able to run Microsoft Excel or Word without Windows.

When shared libraries, or DLLs, as discussed above, are considered, it is very difficult to draw a line between an operating system and an application program. A DLL can be supplied as part of an application program (like the DLLs that come

with Netscape) or as part of Windows (like the four DLLs that Microsoft has indicated make up the Internet Explorer component for Windows), yet they are used by an application program in exactly the same way. It is always possible to take DLLs used by an application program and include them as part of the Windows distribution, much as Microsoft did with the DLLs that were a part of the Internet Explorer 4 distribution when it produced Windows 98.

Combining Programs

As previously mentioned, in this brief I am discussing the combining of two (or more) computer programs to form a single program which replaces the prior programs in the market. When one of those programs enjoys a dominant position in the market, there must be a good reason for combining that program with another, because the separate markets for the two original programs will be reduced or eliminated, and that may reduce competition. It is at times of competition that we have seen the most innovation in software – both the times when it seemed like every major university was developing new operating systems and programming languages to demonstrate interesting concepts, and when commercial developers have competed in the marketplace based on the features of their products. (The best examples of this are the competition between Microsoft and Digital Research in the DOS arena, and Microsoft and Netscape in browser innovations.)

Although this Court was proposing a test for determining whether something was “integrated” as that term was used in the Consent Decree, this Court provided

an excellent starting point for determining whether the combining of two computer programs is tying under the antitrust laws.

> [T]he combination offered by the manufacturer must be different from what the purchaser could create from the separate products on his own. The second point is that it must also be better in some respect; there should be some technological value to integration.

United States v. Microsoft Corp., 147 F.3d 935, 949 (D.C. Circuit 1998). In other words, there must be some synergistic effect from the combining of the programs that can only be achieved by the producer of the two programs combining them. Without such a synergistic effect, the two programs have simply been metaphorically bolted together with no resulting technological benefit for consumers.

Moreover, any improvement must come from the synergistic combination of the two programs, not from improvements to one or both programs that do not stem from their combination. Programs are always being improved, and it is likely that a combination of two programs will use the latest versions of those programs, perhaps versions that have not been available previously. For example, Windows 98 includes a number of improvements over Windows 95 that do not stem from the inclusion of Internet Explorer, such as a new device driver model compatible with Windows NT5, better computer power management, and support for multiple displays and new types of hardware.

There may be benefits from the combining of two programs that are not technological, and such benefits are outside this Court's test. For instance, it may be possible to charge less for the combined program than the sum of the previous

prices for the two programs because it is being distributed in a single box on a single compact disc, but that does not add technological value to the combination.

Perhaps one of the major non-technological benefits that comes from any combining of two programs is that when there is a problem that cannot be readily determined as coming from one of the programs or the other, because they come from a single company you only have to go to the technical support people for that company. There won't be "finger-pointing" between two companies, with the user in the middle. You don't even have to determine which program is causing the problem, since there is only the combined program. There are likely to be more interactions between an operating system and an application program that could cause problems than between two application programs.

The benefit of having a single technical support organization for the combined program is not a technological benefit – it does not result from any synergy between the two programs. And it is something that does not need to be provided only by the software developer, but can be done by an OEM. Often, an OEM will take on the role of a technical support middleman, especially when that OEM has put together a system with software from a number of suppliers to gain a competitive advantage over other OEMs that do not offer that software configuration.

Tautological Benefits

There are some benefits that will always be true when two computer programs are combined. The single technical support organization just discussed is

such a tautological benefit. These should not be considered when applying this Court's test to determine if the combination is a tie under antitrust law unless the synergistic benefit that results from the combination is beyond what would be expected from combining two arbitrary programs. Otherwise, because these benefits are always present when two programs are combined, every combination of programs would meet this Court's test.

Examples of such tautological benefits from combining two computer programs include that the combination can be installed using a single installation program, rather than two separate programs; that the user does not have to assemble the two programs to get the combined program; and not having to support older versions of an operating system in the program that was combined with the operating system because it will always be run with the newer operating system. (All three of these benefits were claimed by Microsoft in the *Caldera* case as benefits that resulted from the combination of improved versions of Windows 3.X and MS-DOS to get Windows 95. Microsoft's attorney, in a hearing before the District Court, stated that having a single installation program was the "most meaningful benefit" of combining Windows and MS-DOS.)

Another tautological benefit particular to combining an application program with an operating system is that a particular shared library that was combined with the operating system is now available to any application program developer using the operating system. This is achieved simply by shipping the shared library along with the operating system. For example, since Microsoft ships its Internet

Explorer DLLs (discussed above) as part of Windows 98, application program developers can use those DLLs' functionality without having to ship the DLLs with their own application programs.

But similarly, if Microsoft were to ship Access, a Microsoft database application program, with Windows, then other application program developers could write their programs to use the Access database system for keeping track of information, knowing that it would always be there because it was included with Windows. The technological benefit does not result from anything special about Access or Internet Explorer, but simply because it was included with the operating system distribution and therefore is available to application program developers.

Another tautological benefit, to the extent that a function is performed by each of the two programs, is that only one copy of the code necessary to perform that function is included in the combination. Microsoft makes this point on page 43 of its brief, where it says "eliminating redundant software code and reducing the amount of such code that must be loaded into memory can be beneficial to users." Of course, libraries that are shared by applications such as the operating system help program or a browser minimize such redundant code. But the benefit will exist whenever the two programs being combined have at least one function whose implementation code can be shared.

If a tautological benefit were enough for a combination of two existing programs to replace those two programs in the market, then there would never be a tying prohibition in computer software. If Microsoft decided to ship Microsoft Office

(its office suite of programs that includes the Word word processor and Excel spreadsheet programs) along with Windows, it could claim that there was a benefit to the user because of having a single program to install everything, and therefore it could never be a tie in violation of the antitrust laws. It is not reasonable to grant such a blanket exemption to the antitrust laws for computer software.

The Value of the Benefit

In the context of the Consent Decree, this Court indicated that the test for "integration" should not consider whether the combination of the computer programs results in an overall benefit to the users of the combined program.

> The question is not whether the integration is a net plus but merely whether there is a plausible claim that it brings some advantage. Whether or not this is the appropriate test for antitrust law generally, we believe it is the only sensible reading of § IV(E)(i).

United States v. Microsoft Corp., 147 F.3d at 950. But this Court should not give a pass under antitrust law to a program that does not produce an overall benefit for consumers. Merely stating a plausible benefit, that may be outweighed by the negative consequences of the combination, should not be enough.

It is important to remember that the combining of two existing, separately-marketed programs so that only the single, combined program is now available to consumers inherently has a detrimental effect when one of the programs has a dominant position in the market. It has the potential to reduce or eliminate competitive markets in the separate programs, and it is competition that has brought innovation to the computer software arena.

When Microsoft decided to combine improved versions of Windows 3 and MS-DOS to produce Windows 95, the market for a compatible DOS was effectively eliminated. Unless there is a special need, most people would rather just install a single, combined program than two programs from separate sources. (This is because of the tautological benefits of a single installation program and of having a single point of contact if there is a problem.) Novell, which was marketing a compatible DOS (DR DOS, originally developed by Digital Research) was left with a greatly-reduced market because it was unlikely that most users of Windows 95 would pay extra and go through the additional effort to replace the DOS portion of Windows 95 with DR DOS. While it is, of course, speculation what would have happened had the separate markets remained for the Windows 32-bit operating system and the DOS 16-bit operating system used in conjunction with Windows (and without Windows, at the time of Windows 95, by many people running programs like WordPerfect or computer games), it is likely that to compete against MS-DOS, Novell would have come up with innovative solutions that would have extended the viability of DOS for many users and may have even made it a better foundation for Windows so that system crashes were less frequent.

One of the greatest drivers of innovation and customer benefit is competition. The combining of two computer programs to form a single program which replaces the prior programs in the market must at least have synergistic benefits stemming from that combination that outweigh the elimination or substantial reduction of the competitive markets for the separate computer programs.

There are other times where combining programs may have detrimental effects that outweigh the benefits of the combination. There have been a number of viruses that have affected users that are running both Windows and Microsoft's Outlook electronic mail program. The "Melissa" virus is one that received a great deal of publicity, but it is just one of many in the past year. Melissa did not affect users who did not have Outlook as their mail program. I use a non-Microsoft email program. To examine how Melissa works, I had a friend email to me a copy that he received and no problems occurred on my computer.

But had Outlook been combined with Windows so that every system had Outlook running, many more people would have been hurt by Melissa or other viruses. Just as a real virus spreads more rapidly through a homogeneous population, so does a computer virus cause more damage when most computer systems have identical software.

While it may be difficult to determine whether the synergistic benefits from combining two computer programs are outweighed by the detrimental effects to aspects like security, in many cases a benefit claimed may be so minor as to make the determination of whether the benefit justifies the combination quite simple. For example, in the *Caldera* case Microsoft claimed that one of the benefits of combining improved versions of Windows and DOS was the pretty cloud pattern that is displayed when Windows 95 is starting. This hid the various screen displays as things were being loaded. But Microsoft's own documentation indicates that this "boot noise obscurer" can cause the computer to hang with some memory managers

from other vendors. Even ignoring the detriment that always exists when previous competitive markets are reduced or eliminated because existing products have been combined, it is difficult to see any net benefit to consumers from having a pretty picture that may cause their computer to hang.

Why Can't Others Combine the Programs?

Under this Court's test, a combination of two computer programs "must be different from what the purchaser could create from the separate products on his own." *United States v. Microsoft Corp.*, 147 F.3d at 949. But *why* the purchaser cannot create the same combination must also be considered.

Sometimes, the combination requires changes to one or both of the existing programs being combined that can only be done by the developer of those programs. But many programs, and especially operating systems, are constructed from modules that can be replaced with alternatives providing the same functionality as well as extended capabilities. Most operating systems, and Windows in particular, allow the addition of device drivers, shared libraries (DLLs), and other modules by suppliers other than Microsoft to support new hardware devices or extend the capabilities of the operating system. Windows even allows the replacement of the user interface with an alternative by changing a line in a configuration file.

Many of the features now present in Windows were first developed as add-ons by other companies, and were later adopted by Microsoft. The initial implementations took advantage of the modular design of the operating system, adding or replacing modules to achieve the desired functionality.

A more likely reason why the purchaser can't do the combination, especially when the purchaser of the two computer programs is an original equipment manufacturer (OEM) who is combining hardware and software for sale to the public, is that the license for a program does not allow the OEM to make any changes to the distribution they receive. For example, an OEM might not be permitted to replace the screen manager (the program that manages computer display, as well as providing a command menu that the user clicks to start a new program), something that may be necessary to provide the capability of clicking on a Web link displayed as an icon on the screen to go to the location associated with that icon. Because of the restriction against modifying the distribution from the operating system vendor, it would not be possible for the OEM to substitute an alternative screen manager from another vendor that provides the same capability, and perhaps some innovative capabilities not available with the operating system vendor's screen manager, to better compete by offering an enhanced product.

In that example, the reason why the OEM can't combine two programs to get the same capabilities is not technical. It is because the operating system vendor will not permit it. And that can't be a satisfactory reason when the operating system vendor is using its market power to get those restrictive license terms.

This is not to say that an OEM which substitutes alternatives from other vendors for modules supplied by Microsoft should be allowed to "pass off" the resulting combination as "Windows," because that would likely be a violation of trademark law. But it is more likely that an OEM who has gone through the effort

of combining modules from other vendors to achieve some enhanced capability would advertise that capability, rather than try to pass off the combination as an unmodified Windows.

Substituting an alternative module for one supplied by Microsoft may not violate copyright law, and certainly not because of any "integrity of the work" argument. The United States recognizes "moral rights" of attribution and integrity only for works of visual art in limited editions of 200 or fewer copies. (See 17 U.S.C. 106A and the definition of "work of visual art" in 17 U.S.C. 101.) A bookstore *can* replace the last chapter of a mystery novel without infringing its copyright, as long as they are not reprinting the other chapters but are simply removing the last chapter and replacing it with an alternative one, but must not pass the book off as the original. Having a copyright in a work does not give that copyright owner unlimited freedom in the terms he can impose. See, for example, *Lasercomb America Inc. v. Reynolds*, 911 F2d 970 (Fourth Circuit, 1990).

Determining the Nature of the Combination

Before discussing a possible procedure for applying this Court's test to see if the combining of two computer programs to form a single program which replaces the prior programs in the market falls within this Court's test, I'd like to discuss some procedures that are not effective.

Such a determination clearly cannot be done at a superficial level. The Windows 95 user interface looks considerably different from that of Windows 3.1. But that may be the result of changes only in the Windows portion of a

Windows/DOS combination, and might not require significant changes to the DOS portion of the combination. This Court's test requires that the combination produce some synergistic result, not that the combination have additional or different capabilities because the programs being combined are improved versions of the existing programs.

In the procedure this Court used in the context of the Consent Degree to determine whether the Windows 95/Internet Explorer combination was "integrated," this Court tried to place that combination on a spectrum ranging from a "non-integrated" Windows 3.X/MS-DOS to an "integrated" Windows 95. That will only work if it is clear that the ends of the spectrum indeed represent "integrated" and "not integrated." But that may not be the case for the end-points used by this Court in its previous decision. There is little in the record leading up to this Court's decision regarding whether Windows 95 is really an integrated product, or whether it is simply upgraded versions of Windows 3.X and MS-DOS operating together as before.

The Consent Decree's acceptance of Windows 95 as permissible predates the first release of Windows 95. There is no evidence that the Department of Justice did a technical analysis of Windows 95 and how it differed from a combination of Windows 3.1 and MS-DOS. But the United States District Court for Utah found that there was a genuine issue whether Windows 95 was really an integrated product, and denied Microsoft's motion for summary judgement on that issue. *Caldera v. Microsoft*, 72 F.Supp. 2d 1295, 1397 (DC Utah 1999). If, in fact, Windows

95 is no more integrated than Windows 3.1 and MS-DOS were, other than being marketed only as a combination, then the endpoints of the comparison spectrum are really the same, and it is impossible to learn anything by trying to place the combination under review on such a zero-length spectrum.

But even if the extent of integration between Windows 95 and Windows 3.1/MS-DOS is different, the determination of the integration of any other combination by placing it on such a spectrum requires three times the effort, since one has to determine the nature of the integration for each endpoint and compare the nature of the integration for the combination under review to those endpoints.

The test sometimes used by Microsoft, that Windows will not boot or in some other manner will not fully work when the Internet Explorer components are removed, is likewise unlikely to properly determine the true nature of the combination. (See *United States v. Microsoft Corp.*, 147 F.3d at 948.)

There are many reasons why Windows will not boot, but the most likely is that when the Internet Explorer capabilities were added to Windows, one or more modules of Windows were replaced with upgraded versions. If the Internet Explorer components that were removed included one or more of these upgraded Windows modules, and the original Windows module are not replaced, it is not surprising that problems would result. This is analogous to replacing a light switch in your home with a dimmer (a module with added capabilities). One would not be surprised if your lights no longer worked if you removed the dimmer but did not put back the original light switch.

Similarly, not being able to perform a function based on a capability introduced in the upgrade of one of the programs being combined does not provide a good test. Since Microsoft has changed the format of its help files so that they require an HTML rendering engine for their display, it is not surprising that they will not be displayable if the HTML rendering engine is removed. If Microsoft wanted to justify the combination of their word processor, Microsoft Word, with Windows, they could have changed the format of the help files to that used by Word. If they had wanted to justify combining their database system, Access, with Windows, they could have stored the system registry, which contains configuration information, as an Access database. If one is clever enough, one can justify the combining of most application programs with Windows, and then show how a capability of the combination is lost when that application program is removed.

It is interesting to note that of the four shared libraries that Microsoft describes as the components of Internet Explorer now included in Windows 98 (see page 31 and the Glossary of Microsoft's brief), only one (MSHTML.DLL) is mentioned as used to provide "Internet Support for ISVs" (see page 42-43 of Microsoft's brief) or the "Windows 98 Help System" (see page 44 of Microsoft's brief).

A Proposed Procedure for Applying This Court's Test

Based on the discussion above, it is possible to come up with a test for determining when the combination of two computer programs produces the synergistic technological benefits this Court considered as key to whether the

combination is permissible. The prior existence of the two programs greatly simplifies this proposed procedure

Again, it is important to recognize that this procedure is only necessary when two existing programs are being combined, one of the programs has a dominant market position, and those two programs will not also be separately marketed. When the original programs remain in the market, especially with the improvements that were made to the programs in the combination, there should be no objection to the combination unless there is an antitrust violation for another reason. Consumers and the market will have the power to decide whether the single combined program or combinations made by consumers or other vendors using one or both of the programs is best, and when there is no longer a market for the separate programs.

As the first step of the proposed procedure, the defendant should state the benefits that accrue from the combination of the existing programs, because the defendant is in the best position to identify those benefits. The benefits do not necessarily have to be the original reasons for combining the existing programs, but can be serendipitous benefits found after the combination has been made.

Having to state the particular benefits from the combination greatly simplifies the work that must then be performed to determine how those benefits are produced. Instead of the plaintiff's expert having to look through millions of lines of source code to opine what benefits there may or may not be in the

combination, that expert can do an in-depth study of how each of the claimed benefits are produced.

Second, the claimed benefits should be examined to determine whether they are benefits that don't directly benefit consumers, non-technical benefits, or tautological benefits, all of which should normally be excluded. For example, if the combination makes it easier for the developers to maintain the program, that should normally be excluded from the analysis because it does not directly benefit consumers. However, if a benefit that would normally be excluded is far greater than would normally be expected when similar programs are combined, then it should be considered.

Finally, the source code for the combination should be examined to determine whether the benefit stems from the combination, or from improvements to one of the existing programs that are being combined. This analysis is greatly simplified by the presence of the existing programs. Commonly-available software development tools, like source file comparison programs, and a review of any change history log or comments for the programs, can be used to determine what produces the claimed benefit.

At this stage, it will also be simple to see if code from the two existing programs have been comingled so that it is impossible to separate one program from another. Because code can always be moved from one module, there must be some technical justification for such code movement.

Also at this stage, any reasons why somebody other than the developer of the combination could not join the separate products to create the same combination can be determined. There must be a technological reason for such inability, and not simply that the developer of the combination does not permit it.

A procedure such as this will determine if this Court's test for permissible combinations has been met:

> [T]he combination offered by the manufacturer must be different from what the purchaser could create from the separate products on his own. The second point is that it must also be better in some respect; there should be some technological value to integration.

United States v. Microsoft Corp., 147 F.3d at 949.

Such a Procedure is Well Within a Court's Competency

This Court has correctly warned against embarking on product design assessments. Because of the limits on their institutional competency, "A court's evaluation of a claim of integration must be narrow and deferential." *United States v. Microsoft Corp.*, 147 F.3d at 949. But that does not mean that a court should give any combination of two computer programs to form a single program which replaces the prior programs in the market a free pass.

It is important to keep in mind that as long as the existing products remain available until there is evidence that there is no market demand for them, any combination of computer programs is permissible as long as it does not violate other principles of antitrust law. A court is not being asked to second-guess product designs in general, but only whether it is an antitrust violation to replace two

computer programs that have previously been marketed separately, one of which has a dominant market position, with a single combination while no longer offering the two computer programs separately. As long as the two programs that make up the combination continue to be offered separately, there is no need to examine their combination.

In other areas of the law, courts are called on to make decisions about computer software designs. In copyright law, under the test first stated in *Computer Associates v. Altai*, 982 F.2d 693 (Second Circuit, 1992), courts must decide whether a portion of a computer program is substantially similar to a copyrighted program due to efficiency considerations or dictated by external factors. In patent law, courts must decide whether the claims of a patent read on a computer program, or whether the program performs substantially the same function, in substantially the same way, getting substantially the same result as the claimed invention.

The procedure I proposed above is similar in complexity to the copyright and patent analyses – perhaps easier because the benefits to be considered have to be particularly pointed out in defense of the combination. In fact, it resembles the abstraction-filtration-comparison test that is the heart of most copyright infringement analysis.

In those cases, the court is not asked to make such determinations without help. Both sides will put forward technical experts for examination by the court to aid it in its determination. Because of our adversarial system, each side will try to

make the issues clear to the factfinder while knowing that they will be corrected by the other side if they misrepresent anything.

In addition, to assist the trial or appellate courts, neutral experts can be used by the courts to help the judges better understand the technology and aid them in making their decisions. The trial judge in *Computer Associates v. Altai* had the benefit of an MIT professor of computer science, supplied under an agreement with both parties. The American Associate for the Advancement of Science has recently established a project to aid in identifying technical experts that could assist the courts – see http://www.aaas.org/spp/case/case.htm. (I am a member of the project's recruitment and screening panel.) Such experts would, of course, not be making decisions, but only assisting judges better understand the issues and helping any decision be not only good law, but also technologically-sound.

Respectfully submitted,

_____

Professor Lee A. Hollaar

CERTIFICATE of SERVICE

I hereby certify that on this 26th day of December, 2000, I served a copy via first class mail of the foregoing **Brief of Professor Lee A. Hollaar as *Amicus Curiae* in Support of Neither Side** to the following:

Phillip R. Malone
Antitrust Division
U.S. Department of Justice
325 Seventh Street, N.W.
Suite 615
Washington, D.C. 20530

Catherine G. O'Sullivan
Chief, Appellate Section
U.S. Department of Justice
Antitrust Division
601 D Street, N.W.
Room 10536
Washington, D.C. 20530

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

Kevin J. O'Connor
Office of the Attorney General of Wisconsin
P.O. Box 7857
123 West Washington Avenue
Madison, Wisconsin 53703-7957

Christine Rosso
Chief, Antitrust Bureau
Illinois Attorney General's Office
100 West Randolph Street, 13th Floor
Chicago, Illinois 60601

Robert S. Getman
359 West 29th Street
Suite G
New York, New York 10001

Bradley P. Smith
Sullivan & Cromwell
1701 Pennsylvania Ave. N.W., 8th floor
Washington, DC 20006

John Warden
Sullivan & Cromwell
125 Broad Street
New York, NY 10004

William Neukom
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Edward J. Black
Computer & Communications Industry Ass'n
666 Eleventh Street N.W., Suite 600
Washington, DC 20001

Robert H. Bork
1150 17th Street N.W.
Washington, DC 20036

Louis R. Cohen
Wilmer, Cutler & Pickering
2445 M Street N.W.
Washington, DC 20037-1420

Donald M. Falk
Mayer, Brown and Platt
1909 K Street, N.W.
Washington, DC 20006

Laura Bennett Peterson
700 New Hampshire Avenue, N.W.
Washington, DC 20037

Paul T. Cappuccio
America Online, Inc.
22000 AOL Way
Dulles, VA 20166

Carl Lundgren
5035 South 25th Street
Arlington, VA 22206-1057

_____

Lee A. Hollaar