It’s Not About the Browser

Microsoft’s Monopoly Derives from Microsoft’s Ability to Control the Windows Interface

Prof. Bruce L. Jacob
Electrical & Computer Engineering Dept.
University of Maryland
College Park, MD 20742
http://www.ece.umd.edu/~blj/
blj@eng.umd.edu
301-405-0432

Prof. Bruce R. Jacob
College of Law
Stetson University
Gulfport, FL 33704
http://www.law.stetson.edu/faculty/jacob/
jacob@law.stetson.edu
727-562-7866

One-Liner

A father/son lawyer/computer-engineer team presents disturbing facts in the Microsoft case that the average person has never heard about, criticisms of the remedies proposed to date, and a novel analysis of the case that points directly to the only remedy that would prevent Microsoft from repeating its illegal behavior in the future.

Abstract

In this article we present an analysis of one of the main threads of the Microsoft “Browser War” and a proposed remedy that we believe is the most effective way to prevent Microsoft from repeating its illegal behavior in the future. Microsoft saw a combined threat to its operating system enterprise from the fusion of the World-Wide Web, Netscape’s popular web browser, and Sun’s cross-platform Java technology. Microsoft was so concerned about protecting its operating system monopoly that it put economic pressure on various corporations to harm Microsoft’s competitors, and it subverted standards of compatibility to undermine the attempts of numerous organizations to increase the ability of computers to interoperate with each other. Reorganizing the company into two parts, akin to the “Baby Bell” break-up, will not prevent future repeats of Microsoft’s more subtle maneuvers. Neither will the settlement agreement of November 2nd, 2001. Instead, we propose an open-standards measure to put Microsoft’s most widely used computer interfaces into the public domain and their specification under the control of an independent body representing the public. This measure is based on the power of the United Stated District Court to enforce the antitrust laws of the United States.

Introduction

As the Microsoft antitrust case nears a close, it is becoming clear that one of the most significant aspects of the case is being overlooked. At issue is Microsoft’s abuse of computer interfaces to harm others, a theme that is not mentioned directly in any of the court records and that is not fully addressed by any proposed remedy to date. All remedies

proposed so far—from Judge Thomas Penfield Jackson's remedies to those proposed in the latest settlement agreements between Microsoft and the United States Department of Justice—fall short of a complete solution to the problem. Neither breaking the company in two nor Jackson's interim provisions nor the watered-down remedies in the proposed settlement of November 2nd, 2001 would prevent Microsoft from behaving in a manner that is effectively identical to its past improper behavior. Moreover, the company's latest operating system release, Windows XP, violates the very same laws, and in the same manner, that the company was charged with breaking in the first place.

We present an analysis of the facts in the case that ties together a string of seemingly unrelated actions. The main issue of the case is not whether Netscape's browser and Microsoft's browser should have equal access to the pre-loaded PC desktop, nor is it whether Microsoft should be allowed to embed browser technology into its operating system. The facts in the case are much more interesting, and far more insidious. Microsoft took on Netscape not because it felt its *browser* enterprise threatened (at the time, Microsoft had no browser enterprise whatsoever); Microsoft took on Netscape because Netscape's browser had the potential to make Microsoft's *operating system* enterprise irrelevant. Court records [1,2,3] show that Microsoft anticipated a combined threat against its operating system enterprise from the fusion of Netscape's popular web browser, Sun's cross-platform Java technology, and the open-standards communication protocols of the World-Wide Web. Microsoft diverted this threat by intentionally undermining the attempts of numerous organizations to make computers more compatible with each other. Where other companies and groups proposed and implemented open standards, Microsoft implemented its own proprietary and incompatible standards, and through its monopoly status was able to divert attention from and effectively neutralize several key software interfaces proposed by others. In this particular instance, had Microsoft failed, and had the Java/Netscape vision achieved its potential (as seemed very likely at the time), computer users would have had the option to run whatever software they wanted, using whatever operating system and hardware platform they cared to use, rather than be tied to the Windows/PC platform.

It is crucial to understand that Microsoft maintained its monopoly status not only by bullying other companies (which is obviously illegal once a company is found to be a monopoly) but also by manipulating computer interfaces—both its own and those of others. It is this latter behavior that has been overlooked and which is not fully addressed by any remedies proposed to date; we will show that this manipulation of computer interfaces was just as effective a method for extending Microsoft's monopoly as bullying. Our conclusion is that the only way we can prevent future misconduct on the part of Microsoft is to require the company to not only disclose to the public its internal interfaces but also relinquish its rights to those interfaces and, furthermore, to adhere to existing open standards.

Note that this remedy goes one step further than Judge Jackson's interim provision that Microsoft fully disclose the details of its Windows operating system's interface[1]. Microsoft has demonstrated in the past that it will target selected competitors and implement changes to the Windows interface that render the software of those targeted companies suddenly incompatible with the newest version of Windows. The ultimate effect of this behavior is obviously to reduce the market penetration of the targeted competitor's software. This behavior, we believe, violates the Sherman

Act and must be stopped. Simply divulging the details of the Windows interface will not curb Microsoft's behavior, as we will discuss in more detail later. The only way to stop Microsoft's illegal behavior is to take away the company's right to manipulate the Windows interface at will; this is an interface that is so widely used that it has become the *de facto* standard for writing software in today's world-wide computer network, and any change to this interface affects potentially millions of people. Because Microsoft has a clear history of manipulating this interface to intentionally harm competitors, the company's behavior must be stopped.

Sidebar: A Brief History of the Microsoft Case

In 1998 the Justice Department of the United States filed an antitrust action against the Microsoft Corporation, alleging that Microsoft was an illegal monopoly and was engaging in activities that violated the Sherman Antitrust Act. A number of states also filed federal antitrust actions against Microsoft. Those actions were joined with the suit by the Justice Department, and United States District Judge Thomas Penfield Jackson, of the United States District Court in the District of Columbia, was assigned to the case. Efforts were made to bring the government and Microsoft to a settlement, but those efforts failed. A trial was held, and Judge Jackson issued *Findings of Fact* on November 5, 1999 and *Conclusions of Law* on April 3, 2000, determining that Microsoft did violate the Sherman Act. Microsoft requested a hearing on the issue of the remedies that should be imposed, but Judge Jackson, disgusted with the behavior of Microsoft's defense during the trial, said a hearing was not necessary—because many of Microsoft's witnesses had perjured themselves or presented falsified evidence [24], and Jackson felt it was clear that the same would be true for any of Microsoft's future witnesses. Jackson issued his Final Judgment on June 7, 2000, and that judgment contained a bombshell—Jackson issued a structural remedy that split Microsoft into two corporations: an operating systems business and an applications business, in addition to other remedies.

Microsoft appealed to the United States Court of Appeals for the District of Columbia. On June 28, 2001, that Court issued its opinion in the case. The Court upheld that portion of Judge Jackson's decision finding Microsoft guilty of violating the antitrust laws of the United States, but the Court modified the remainder of Judge Jackson's decisions. The Court stated that Jackson had erred in not providing Microsoft a hearing on the issue of remedies. Also, Jackson had made remarks to the press during the case which showed a bias against Microsoft on his part. The case was remanded to the District Court, and a new judge was to be assigned to the case to re-determine the remedies to be imposed against Microsoft.

Microsoft sought review of the decision of the Court of Appeals upholding the finding that Microsoft was an illegal monopoly, in violation of the Sherman Act. The Supreme Court, in October, 2001, rejected Microsoft's request.

---

1. Throughout this article we will be dealing with computer interfaces; when we use the term "interface" we mean "internal" computer interfaces, which should not be confused with the "user" interface. The user interface is the arrangement of words and diagrams that is drawn by the computer on a computer screen and with which a person interacts; the article is not about this type of interface. We will define in more detail what we mean by the term "internal" interface as this article progresses.

The case is now before Judge Colleen Kollar-Kotelly of the District Court. She appointed a mediator in October, 2001 to help the parties try to settle the case. Within a month, the Justice Department and nine of the states entered into a proposed settlement agreement with Microsoft while nine others of the plaintiff states refused to sign the agreement. Judge Kollar-Kotelly will have to decide whether to approve the settlement and will also have to resolve the pending suits by the nine states that have not joined in the agreement. A date of March 4, 2002 has been set as the beginning date for the hearing to determine what remedies should be imposed against Microsoft.

A Primer on the Computer's Internal Interfaces

Definitions and Clarifications

A computer interface is a language. It is a set of rules by which computer components, both hardware and software, interact. It is a contract between specific components that specifies the syntax and semantics of any and all interactions that involve those components. Interfaces are found at all points of contact between computer components, including application-to-application, application-to-operating-system, software-to-hardware, and hardware-to-hardware:

- Application-level interfaces dictate the level of compatibility and interaction between different software applications. Example: for a Netscape plug-in to work correctly, it must use the correct application-level interface to interact with the browser.
- The interface between computer programs and the operating system, often called the application programming interface (API), determines what services a program may request of the operating system. Example: for a computer program to work correctly on the Windows operating system, it must use the Windows API, the interface into the Windows operating system.
- Interfaces between software and hardware, called instruction sets, determine what operations a program can perform in hardware. Example: for software to run correctly on an Intel-based computer, it must use the x86 instruction set, the interface that the hardware understands.
- Hardware interfaces that connect devices—including chip-to-chip interconnects such as PC-100 SDRAM, DDR SDRAM, Rambus Channel, and the Intel P6 bus, as well as peripheral bus or networking protocols such as SCSI, PCI, USB, and Ethernet—specify how hardware components interact and, in the case of peripheral and networking protocols, also specify how software is to use the hardware to communicate. Example: for a computer to successfully talk to a USB device, such as a mouse or printer or disk drive, the operating system and the device must understand the USB interface, and the computer must have a USB connector.

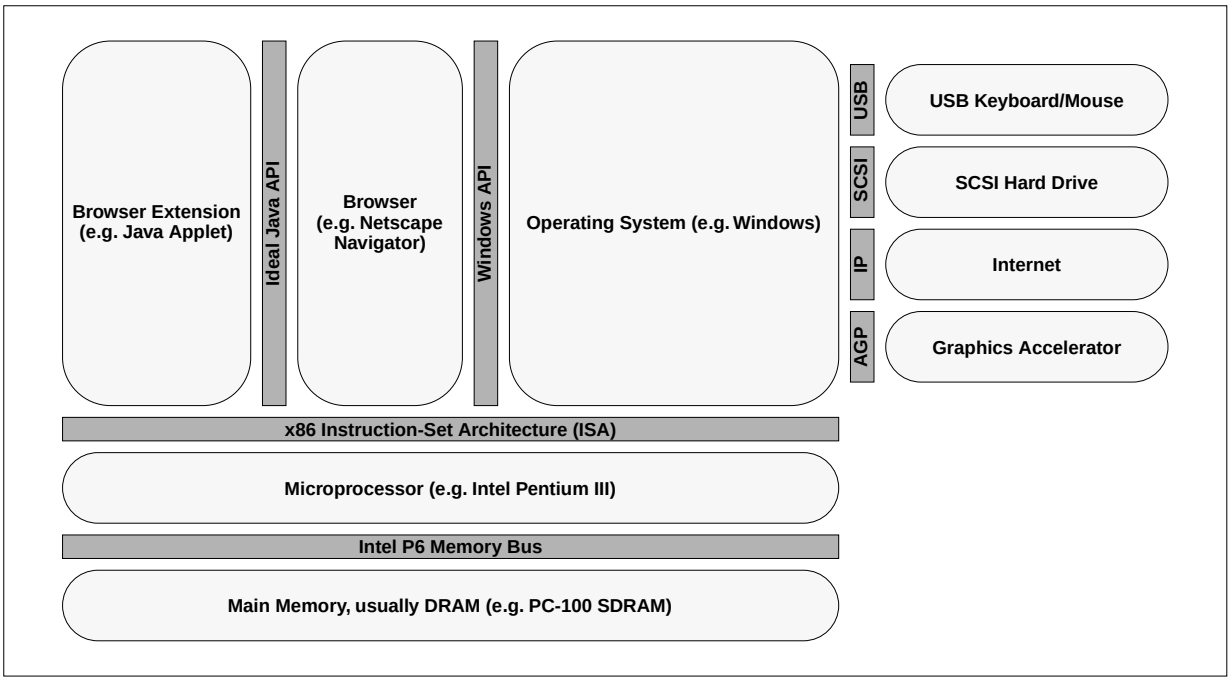

Figure 1: The various interfaces found in a typical personal computer. Computer components are shown in light grey; the interfaces through which they interact are shown in darker grey. Components that are not separated via an interface cannot communicate directly with one another. For example, the keyboard cannot write directly to the Internet— typed messages must go through the operating system. The interface between the browser and browser extension is labeled "ideal" because it does not exist exactly as drawn. Microsoft ensures that all software running on a PC must depend at least partly on the Windows API; Microsoft allows no piece of software to even partially supplant the Windows API, otherwise the importance of Windows might diminish.

Figure 1 illustrates examples of these interfaces as found in a typical personal computer. The figure shows a browser running on an operating system and a Java applet running within the context of the browser. The Java API between the browser and the browser extension is an application-level interface that allows a software program to use the services of another software program. The Windows API is the interface through which software programs request services from the operating system that they cannot perform by themselves. For example, the browser does not have direct access to the Internet and must ask the operating system to send network packets out on its behalf; typical programs do not have direct access to the keyboard and mouse and must ask the operating system for user input; and only the operating system can read and write the hard drive. The instruction-set architecture is the interface between hardware and software and consists of very simple requests such as *add the following numbers*, or *multiply the following numbers*, or *store this value to memory*, etc. As the figure shows, all software has direct access to the microprocessor through this interface; for example, while the browser is running it interacts directly with the hardware until it makes a request that only the operating system can service (such as sending a network packet or retrieving user keystrokes from the keyboard or writing a file to the disk). Lastly, hardware-to-hardware interfaces such as the Intel P6 memory bus allow hardware components to interact directly, and peripheral busses such as USB (Universal Serial Bus) and networking protocols such as IP (the Internet Protocol) allow the operating system to talk

to hardware devices connected to the computer and other computers across the Internet, respectively. One thing to note is that the operating system plays a critical role within the system, as it represents the crossroads of all activity; very little happens in a computer that is not under the direct control of the operating system.

Note that the Java applet is shown executing entirely within the context of the browser; in particular, the figure shows the Java applet executing without having to use the operating system directly. This is an ideal organization; in reality, due to the particulars of Microsoft's Java support, a Java applet is at least partly dependent on the Windows operating system. Nonetheless, this serves as a good example of what interfaces do and how they do it. Interfaces are designed to separate components from each other and thus simplify the business of developing complex computer applications. If the operating system supports a particular interface or device, then applications can use that interface or device without much additional effort. For example, if programmers want to use a certain graphics coprocessor, and the operating system supports that device, then the programmers can develop software that uses the graphics coprocessor without having to incorporate its interface into their software because that interface is already incorporated in the operating system. The flip side of this is that if the operating system does *not* support that particular device, the programmers are usually prevented by the operating system from interacting with the device at all—as the diagram shows, software programs are separated from hardware devices by the operating system.

As we have said, a computer interface (as opposed to a graphical user interface) is a language, and as with any other type of language, if either the syntax or semantics are disobeyed, even minutely—which can often happen by writing imperfect software code or building imperfect hardware—then communication between components fails. Similarly, if two people are speaking the English language, but one occasionally uses the wrong tense for verbs or forgets to use articles or occasionally speaks in another language with which the second speaker is unfamiliar, then the second speaker will have enormous difficulty with the conversation.

The problem of violating computer interfaces is more serious than speaking human languages improperly, and this is why we have described computer interfaces as contracts as well as languages. Because computers and computer components tend to be generally intolerant of errors, it is the case that even simple violations of an interface, such as swapping the order of two control statements (analogous to switching the order of subject and verb in a sentence), typically have devastating effects, such as a computer program running erratically or not at all, or perhaps the entire computer reaching a state that requires a reboot. To avoid such miscommunications, designers are very careful to be both precise and thorough when they write up the specification of an interface; such specifications are very complex and contain, among other things, all of the valid command codes of the interface, any expected responses, and semantics attached to each valid command code. To re-define the interface is to make changes to its specification. An *open* interface (often called an open standard) is one in which all of the details of the interface are made publicly known, often through documentation freely available on a widely publicized website. A *public* interface or public standard is one whose specification is open and controlled by an independent public organization (more on this later). A *de facto standard* interface is one that is widely used; *de facto* standards may be public, and they are usually open, but this is not always the case. The Windows API is an example of a *de facto* standard that is neither public nor open.

An interface is an abstract entity and must not be confused with the software codes or hardware mechanisms that implement it. Software codes and hardware mechanisms are concrete entities; they can be created, destroyed, held in the palm of your hand. Interfaces are like the terms of a contract—though you can define them on a piece of paper, burning that paper does not make the terms of the contract (or the details of the interface) disappear. Most importantly, because an interface is not the same thing as an implementation of that interface, one does *not* have to divulge the details of an implementation to divulge the details of an interface, and disclosing the details of an interface in no way compromises the security of any implementation of that interface. In the case of Microsoft's Windows operating system, the company can easily divulge the full details of the operating system's interface without having to make public the details of its implementation—i.e. the software code of Windows that implements the interface, and it would be impossible for even a computer expert to infer any details of Microsoft's software code just by looking at the interface specification.

A good analogy is the steering interface of a car. The car has a steering column that, if used properly, turns the wheels of the car and thereby changes the direction of car's movement. The steering column's interface is relatively simple and can be described with the following specification.

> STEERING INTERFACE: If the steering column is twisted clockwise, the car will turn to the right; if the column is twisted counter-clockwise, the car will turn to the left. How far the column is twisted determines how sharply the car turns. The car will continue to turn from a straight course until the steering column is returned to its original rest position in the center.

Though this may not sound much like a language or contract, it is: This interface specifies exactly what actions a driver may perform and exactly what the car will do in response to those actions.

This interface has numerous potential implementations. An obvious one is a wheel attached to the steering column; a driver grips the edge of the wheel to get better leverage on the column. Another implementation is a set of handle bars, such as one would find on a motorcycle. Another implementation is a rudder sticking out at right angles to the steering column. An *ad hoc* implementation similar to the rudder is a pipe wrench gripping the steering column directly. One could create a "power steering" implementation by attaching an electric drill to the steering column, such that pulling the drill's trigger twists the steering column, and the direction of the twist is determined by a setting on the drill. The alternatives are endless.

The important point to note is that each of these implementations is a concrete object and involves some degree of creativity and invention to construct. The interface specification applies to all the implementations and is embodied by them, but it is not a concrete thing. One can divulge the details of the interface without mentioning or implying any particular implementation of that interface (as is the case with the specification given above); and it would be impossible for even an expert engineer to infer any details of any particular steering implementation just by reading the interface specification (as is also the case with the specification given above).

In this analogy, the interface specification given above would correspond to the Windows API, and any one of the interface implementations (e.g. the rudder example) would correspond to the software code that implements the Windows operating system. One writes code to build a program that supports or uses a computer interface, just as one would build a steering wheel or set of handle bars to use the steering-column interface. Just as the paragraph above detailing the steering interface gives no hints as to how one would build a particular steering wheel, knowing the full details of the Windows API gives no hints as to how the Windows operating system is built.

The analogy extends even further. If the interface's specification were to change, then potentially all the implementations would be affected, meaning that they may or may not implement the new interface. For example, if the car's manufacturer redefined the interface to heat or cool the steering column (heating up the metal turns the car right; cooling it down turns the car left; magnitude of temperature change determines sharpness of turn; car moves in straight line if temperature of steering column is 75º …), then all of the steering implementations above would suddenly no longer work with new cars using the new interface. Though the example might seem a bit whimsical, it uncovers one of the fundamental dangers of widely-used interfaces, as detailed in the next section.

The Impact That Interfaces Have on the High-Tech Industry

Because of computer interfaces and the highly structured ways in which computers interact, there exist today modes of corporate behavior that were not possible a decade ago. It is now possible for a powerful corporation to exploit the way computers are made to injure others who build computers and computer-related products. Because the economy is heavily intertwined with the high tech industry, such actions today can adversely affect a large number of corporations, whereas such actions might have been little more than annoyances ten years ago. This mode of corporate behavior is something akin to twisting the rules (i.e. the specifications of widely-used computer interfaces) so as to harm those who have no choice but to follow those rules.

As we have indicated, internal interfaces are the languages, or sets of rules, that specify how computing components interact. If either side disobeys the rules (e.g. attempts to use a control signal that does not exist or is not recognized by the other component), then it is as if one side is speaking gibberish, and communications between components fails. Therefore it is of utmost importance that both sides adhere very strictly to the interface's specification. When an interface changes—for example, when it is time to "upgrade" the interface to include features that cannot be provided through any other means—then all products previously compatible with that interface must be redesigned to comply with the new specification, else they become out-of-date and quickly lose market share. Redesigning a product for a new interface specification is a substantial undertaking: Such redesign can require enormous amounts of money, time, and engineering effort, and oftentimes a company will simply give up on a product line rather than update it to comply with the latest interface specification, simply because of the cost involved.

As the members of the Court of Appeals mention in their opinion, "Once a product or standard achieves wide acceptance, it becomes more or less entrenched." [1: p. 12] It is therefore extremely important to prevent that entrenched standard's misuse. We believe this is fundamental if we are to prevent monopolistic abuses in the high-

tech industry, because through the manipulation of computer interfaces a monopoly can do much more damage than it could by simply offering competing products. These interfaces have wider impact because not only are end-products and therefore end-consumers dependent on them, but those hardware and software vendors that build compatible end-products are dependent on them as well. Modifying an interface makes potentially all products based on that interface instantly obsolete—because, as soon as the modification is made, those products affected by the changes no longer conform to the current interface, and an interface is ultimately more influential in its scope than any particular product that embodies it … provided the new interface catches on. Because a monopoly is capable of successfully creating and promoting new interfaces at will, a monopoly controlling a widely used interface can, by manipulating that interface, dominate its direct competitors, its indirect competitors, and even those corporations that provide complementary products and do not compete with it in any way.

Given this depiction of the importance of interfaces and the economic impact of their re-definition, it is easy to understand that if a company is allowed to control an important interface, it can very easily change the shape of the computing world. Just such a situation is quite possible because many interfaces are proprietary, and therefore the developer of an interface can change its specification at will. Often, the owner of a proprietary interface can effectively prevent any other company from building or marketing products that use the interface: While interfaces are not patentable, many companies obtain wide-reaching patents for their implementations of those interfaces that make it difficult for any other company to build its own implementation of the same interface without infringing on the first company's patents. This is an unfortunate situation: An analogy would be the "developer" of the English language preventing others from speaking the language, or forcing them to pay royalties to do so.

In most instances, the owners of interfaces do not rapidly change the interface specifications, nor do they charge outrageous licensing fees or single out individual corporations and prevent them from using the interface. If a company were to do any of the above, other companies would most likely realize that dealing with this particular interface is more trouble than it is worth, and they would go find another interface or develop their own. However, if the interface in question is the *de facto* standard, if it is used by all other companies in the same industry, if support for this interface is expected by the end-user, and if there exists no practical alternative, then a company would be in serious jeopardy if it were to run afoul of the owner of that interface.

This is exactly the situation in which many companies have found themselves. The interface in question is that of the Windows operating system, owned by Microsoft. Windows is the *de facto* standard of Intel-compatible operating systems:

> Every year for the last decade, Microsoft's share of the market for Intel-compatible PC operating systems has stood above ninety percent. For the last couple of years the figure has been at least ninety-five percent, and analysts project that the share will climb even higher over the next few years. [2: ¶35]

Any software company that wants a significant market share must write its software for Windows. Any hardware company that wants a significant market share must design its hardware to be interoperable with Windows. Perhaps

the most important point of all is that the software companies know this, the hardware companies know this, and the officers of Microsoft obviously knew it as well.

The situation is particularly trenchant as it applies to alternative technologies—i.e. possible competitors to Windows. Because such an enormous portion of the world's computer users are dependent on Windows, any technology with the potential to compete with Windows must also be compatible with Windows, otherwise no significant number of computer users would ever use it (or even know it exists). Through its ability to change the Windows API, Microsoft has the power to determine what software is compatible with Windows and what software is incompatible. Because of this, Microsoft can target any potential competitor and make its product incompatible with Windows for a long enough period of time to make consumer interest in that product wane; at that point, the product poses no more threat to Microsoft's monopoly. This article will describe Microsoft's use of that power to eliminate the potential threats of Netscape and Java.

The Facts of the Microsoft Case and Their Implications

Microsoft has a history of manipulating interfaces so as to harm other companies and maintain its monopoly status. Microsoft changed the Windows API substantially and rapidly so that other companies failed to keep up with the changes and so lost market share (example: IBM OS/2); Microsoft created small-scale changes to its Windows API so as to disable the software of targeted companies (examples: Netscape, Apple, RealNetworks, America On-Line); Microsoft divulged the details of its interfaces selectively so that the software products of targeted companies would become incompatible with Windows and remain so until Microsoft chose to divulge the details to the targeted companies (example: Netscape Navigator); and Microsoft watered down open or public interfaces by offering incompatible proprietary alternatives and using its monopoly status to lure (and in some cases force) developers away from the standard interfaces (examples: Java, HTML). The rest of this section gives the details of these examples.

The effect of its manipulation of interfaces has been to destroy any newly developed, innovative software application that has the potential to compete with Microsoft's software. Because Microsoft is a monopoly, such tactics are illegal, and therefore Microsoft's practice of manipulating interfaces must be stopped.

The Netscape/Java Threat

In this article we discuss only one of the series of actions taken by Microsoft to protect its software monopoly. The opinion of the Court of Appeals and the *Findings of Fact* of the District Court contain many more examples, but this one is particularly enlightening because it illustrates Microsoft's extensive monopolistic activities concerning computer interfaces. The saga begins with a description of how technological advances tend to occur in the high-tech industry:

> The software industry in general is characterized by dynamic, vigorous competition. In many cases, one of the early entrants into a new software category quickly captures a lion's share of the sales, while other products in the category are either driven out altogether or relegated to niche positions. What eventually

> displaces the leader is often not competition from another product within the same software category, but rather a technological advance that renders the boundaries defining the category obsolete. These events, in which categories are redefined and leaders are superseded in the process, are spoken of as “inflection points.” [2: ¶59]

In common parlance, these “inflection points” are “paradigm shifts” during which an entrenched company can be blind-sided by a new technology that renders the company’s particular segment of the industry unimportant. When this occurs, the company’s dominance of that now-unimportant segment becomes irrelevant: Despite their position, the company becomes marginalized along with their segment of the industry. Given that only ten years ago, less than one household in one hundred recognized the word “Internet,” yet today the Internet is given credit for the economic revival of the 1990’s, it becomes clear that the rise of the Internet represents one of these paradigm shifts.

> The exponential growth of the Internet represents an inflection point born of complementary technological advances in the computer and telecommunications industries. The rise of the Internet in turn has fueled the growth of server-based computing, middleware, and open-source software development. Working together, these nascent paradigms could oust the PC operating system from its position as the primary platform for applications development and the main interface between users and their computers. [2: ¶60]

Internet-based computing was seen by Microsoft as a potential threat to the Windows operating system because the Internet represents the ultimate in computer compatibility: Any computer that speaks the language of the Internet—its interface, the Internet Protocol (IP, see Figure 1)—can speak to any other computer that speaks the same language, regardless of the computer’s hardware class or instruction set or operating system controlling it. Such ease of interoperability posed a threat to Microsoft because if computers became more interoperable, especially if Internet-based computing allowed users easier access to run programs that interested them by making them available on any machine in the world that was Internet-compatible, this would make one’s choice of underlying operating system unimportant. Any operating system would suffice, and competition among vendors of these systems would take place on a level playing field.

The primary question is *how*? How could a computer user take advantage of the Internet to run applications without regard to the particular operating system on his or her computer? The answer lies in the operating system’s interfaces: Offering similar interfaces via some channel other than the operating system would eliminate the importance of using any particular operating system.

The channel that became available was the combination of Netscape and Java.

By 1994, Netscape was becoming synonymous with the Internet, the company’s browser was used by nearly all those who explored the World-Wide Web, and the browser ran on nearly every type of computer—not just those using the Windows operating system. Due to intense interest in all things Internet-related, many consumers were buying their first PC just to see what all the fuss was about. For many consumers, Netscape’s browser was their portal to the networked world; it provided browsing, access to newsgroups, and email—and many consumers used their PCs to do little more than that. These consumers were in effect insulated from the underlying operating system; because their

primary interaction with the computer was through Netscape's browser, they could have been using any other operating system and would have been oblivious to that fact.

At the same time, Sun Microsystems was promoting a new development environment called Java that promised a "write once, run anywhere" level of compatibility. In other words, a computer programmer could write a program using Java and fully expect it to run on any computer anywhere, whether it was a Windows computer, an Apple computer, a Unix computer, or any other breed on which Java ran. In general, only those applications written for Windows run on Windows computers, only those applications written for Apple computers run on Apple computers, etc. Because Netscape ran on Windows, it was immediately available to that 90% of the consumer PCs that used Windows, and this widespread availability was one of the main factors that helped it succeed. For Sun's Java environment to succeed, it would need to run on Windows too, but, because Java was an obvious threat to Windows in that its goal was to make the choice of underlying operating system immaterial, it was unlikely that Microsoft would ship Java technology with the Windows operating system; moreover, Microsoft executives recognized early on the possibility that Java could erode Microsoft's position [2: ¶75–76].

In May 1995, Netscape promised to integrate Java into its browser. This provided Java direct access to every machine on which Netscape's browser ran, and because of the popularity of the Internet, the Web, and Netscape's browser, this represented a large number of machines that would potentially run Java applications. For developers of computer programs, this meant that programs written in Java would run on nearly every desktop, laptop, and server computer in the world, and this would constitute an even larger market than the market for Windows desktops alone. In addition, Java incorporated abilities that supported computing over the Internet, including the ability to easily download new programs and run them locally, which would enable low-cost, effortless software distribution over the network (one small example of this is the running of Java "applets" that appear on many websites). The increased market for software and the low-cost distribution channel for that software made this environment an exciting potential for program developers.

If the Netscape/Java environment ever reached its full potential, program developers would be able to reach larger markets than before, yet with less cost overhead than before. Doing so would also free consumers to choose whatever operating system they wished, provided it supported Netscape's browser, which, as mentioned, already ran on nearly every operating system then in existence. This scenario would be exactly the kind of "inflection point" that would diminish the importance of Windows and displace Microsoft from its position as software market leader. Bill Gates saw this potential as soon as Netscape and Java merged:

> In late May 1995, Bill Gates, the chairman and CEO of Microsoft, sent a memorandum entitled "The Internet Tidal Wave" to Microsoft's executives describing Netscape as a "new competitor 'born' on the Internet." He warned his colleagues within Microsoft that Netscape was "pursuing a multi-platform strategy where they move the key API into the client to commoditize the underlying operating system." [2: ¶72]

Netscape's "multi-platform strategy" was simply the fact that the browser ran on nearly every operating system existing at the time, not just Windows. This evidently worried Gates because it might make alternative operating systems attractive to consumers. The "key API" was the operating systems API being offered by Java, which could displace the Windows API, and "the client" into which it was being moved was Netscape Navigator itself. Doing so would enable computer users to run Windows or Windows-like applications from within Navigator, possibly over the Internet. Because Navigator ran on nearly every operating system in existence at the time, this would "commoditize" or make immaterial the underlying operating system: Any operating system would be a viable substitute for any other, provided only that it ran Navigator.

Obviously, if one could run Windows or Windows-like applications on any operating system one cared to use, this would weaken the appeal of Windows. Users would perhaps start to shop around for operating systems using criteria such as ease of use, or compatibility with other systems, or system reliability (e.g. relatively infrequent crashes). A competitor with a product easier to use than Windows, or more reliable than Windows, could have taken away much of Microsoft's business and caused a reduction in the sales of Windows.

Microsoft's Actions to Remove the Threat

In response, Microsoft took a number of complementary steps to eliminate the threat posed by the merger of Netscape's browser and Sun's Java technology. All but the first of these steps later would be found to have been illegal, when the courts declared Microsoft a monopoly. It is therefore important that any remedies decree should address these steps.

Microsoft first attempted to dissuade Netscape from allowing its browser to execute other applications, Java or otherwise [2: ¶79–87]. When this failed, Microsoft ran a campaign to erode Netscape's dominance in the browser market, subvert the ability of Java to run on all platforms equally, and weaken the compatibility levels that made the World-Wide Web so popular:

1. Microsoft created an alternative browser to Netscape's, called Internet Explorer.

2. Microsoft brought pressure to bear on OEM (Original Equipment Manufacturer, e.g. Dell, Gateway, etc.) and IAP (Internet Access Providers, e.g. America On-Line, Compuserve, etc.) channels, to ensure that they used Microsoft's browser and not Netscape's.

3. Microsoft changed its Windows API to make existing software incompatible; in particular, the API for its Windows95 release was structured so that Internet applications, e.g. Netscape's browser, would no longer work unless they used the new interface.

4. Microsoft selectively disseminated the details of its Windows API changes; in particular, Microsoft withheld key information about the Windows95 Internet-related interface from Netscape, which delayed the release of

Netscape's Windows95-compatible browser until after the release of Microsoft's Windows95-compatible browser.

5. Microsoft subverted the Java interface standard, by creating a Java implementation for Windows that did not comply with the Java standard and thus undermined the portability of Java applications to and from the Windows operating system.

6. Microsoft attempted to subvert the HTML interface standard, by adding its own incompatible extensions to HTML (Hyper-Text Markup Language, the format of documents posted on the World-Wide Web) and building its own extensive web portal using this non-compliant HTML, thus ensuring that the web site would fail to work correctly for anyone using any browser other than Internet Explorer.

The following sections discuss each of these actions in more detail.

Microsoft created an alternative browser to Netscape's. Microsoft only began to develop a web browser after it became clear that Netscape would not abandon its efforts at making a cross-platform program that was powerful enough to run applications directly (i.e. via the incorporation of Java technology). Microsoft knew that software developers would only write applications, or programs, for Netscape/Java if they believed that the Netscape/Java combination would emerge as a new widespread standard for Internet-based computing. Therefore, Microsoft sought to prevent the Netscape/Java combination from succeeding in the marketplace. Microsoft's initial goal in building Internet Explorer was simply to divert attention away from the Netscape/Java platform: If users believed Internet Explorer to be equal to or better than Netscape's browser, then Netscape would never reach the level of a widespread standard [2: ¶133].

To reach this goal, Microsoft knew that it had to capture at least half of the browser market. "From 1995 onward, Microsoft spent more than $100 million each year developing Internet Explorer," [2: ¶135] and it spent an additional $30 million per year marketing it [2: ¶140]. Moreover, Microsoft never charged a single cent for its product: It offered the browser without license fee, either from users or Internet Access Providers (such as AOL), and, beyond offering the browser for free, "Microsoft actually paid AOL a bounty for every subscriber that it converted to access software that included Internet Explorer instead of Navigator." [2: ¶139]

Had Microsoft stopped at simply creating an alternative browser, it is likely that the company would have avoided legal trouble. The problems started when Microsoft used other avenues besides direct competition to win the browser war:

> Decision-makers at Microsoft worried that simply developing its own attractive browser product, pricing it at zero, and promoting it vigorously would not divert enough browser usage from Navigator to neutralize it as a platform. They believed that a comparable browser product offered at no charge would still not be compelling enough to consumers to detract substantially from Navigator's existing share of browser usage. This belief was due, at least in part, to the fact that Navigator already enjoyed a very large installed base and had become nearly synonymous with the Web in the public's consciousness. If Microsoft was going to raise Internet Explorer's share of browser usage and lower Navigator's share, executives at Microsoft believed

> they needed to constrict Netscape's access to the distribution channels that led most efficiently to browser usage. [2: ¶143]

Microsoft brought pressure to bear on OEM and IAP channels. Because consumers rarely customize their PC desktops to the point of replacing software that comes with the computer by some other software that performs the same function, having one's software pre-installed by an OEM onto a computer is one of the most direct and cost-effective means to get consumers to use one's software. The other primary channel, at least for Internet-related software, is for Internet Access Providers to bundle that software with their product. In the case of a browser, the browser would be the primary access tool to the IAP's content. The District Court agreed that "no other distribution channel for browsing software even approaches the efficiency of OEM pre-installation and IAP bundling. The primary reason is that the other channels require users to expend effort before they can start browsing." [2: ¶145] Therefore, the OEM and IAP channels were identified by Microsoft as the most important conduits to close down for Netscape if Microsoft were to ensure the failure of Netscape's browser.

> Microsoft achieved this feat [(closing down the OEM channel)] by using a complementary set of tactics. First, it forced OEMs to take Internet Explorer with Windows and forbade them to remove or obscure it — restrictions which both ensured the prominent presence of Internet Explorer on users' PC systems and increased the costs attendant to pre-installing and promoting Navigator. Second, Microsoft imposed additional technical restrictions to increase the cost of promoting Navigator even more. Third, Microsoft offered OEMs valuable consideration in exchange for commitments to promote Internet Explorer exclusively. Finally, Microsoft threatened to penalize individual OEMs that insisted on pre-installing and promoting Navigator. Although Microsoft's campaign to capture the OEM channel succeeded, it required a massive and multifarious investment by Microsoft; it also stifled innovation by OEMs that might have made Windows PC systems easier to use and more attractive to consumers. That Microsoft was willing to pay this price demonstrates that its decision-makers believed that maximizing Internet Explorer's usage share at Navigator's expense was worth almost any cost. [2: ¶241]

Microsoft knows that because of the popularity of the Windows operating system, the company has substantial control over what the OEMs pre-load and what they do not pre-load. "Microsoft knows that OEMs have no choice but to load Windows, both because it has a good understanding of the market in which it operates and because OEMs have told Microsoft as much." [2: ¶55] "Without significant exception, all OEMs pre-install Windows on the vast majority of the PCs that they sell," [2: ¶54] and Microsoft helps to ensure this "by advising OEMs that they will be charged a higher price for Windows unless they drastically limit the number of PCs that they sell without an operating system pre-installed." [2: ¶58] In general, Microsoft "charges different OEMs different prices for Windows, depending on the degree to which the individual OEMs comply with Microsoft's wishes." [2: ¶64]

Microsoft used this power to force OEMs to pre-load only Microsoft's browser, Internet Explorer, despite the fact that many OEMs and their customers preferred Netscape's browser, Navigator. Microsoft's first tactic, by making select changes to the Windows API, was to make it technically impossible to remove Internet Explorer from the operating system. Further, Microsoft built the operating system in such ways as to override the user's choice of "default browser" in many circumstances and to open up security holes for users who chose to use Navigator [2:¶155–198]. Some examples:

> [I]n late 1995 or early 1996, Microsoft set out to bind Internet Explorer more tightly to Windows 95 as a technical matter. The intent was to make it more difficult for anyone, including systems administrators and users, to remove Internet Explorer from Windows 95 and to simultaneously complicate the experience of using Navigator with Windows 95. As [Microsoft official] Brad Chase wrote to his superiors near the end of 1995, "We will bind the shell to the Internet Explorer, so that running any other browser is a jolting experience." [2: ¶160]
>
> Microsoft's engineers … [made] Windows 98 override the user's choice of default browser in certain circumstances. As shipped to users, Windows 98 has Internet Explorer configured as the default browser. While Windows 98 does provide the user with the ability to choose a different default browser, it does not treat this choice as the "default browser" within the ordinary meaning of the term. Specifically, when a user chooses a browser other than Internet Explorer as the default, Windows 98 nevertheless requires the user to employ Internet Explorer in numerous situations that, from the user's perspective, are entirely unexpected. As a consequence, users who choose a browser other than Internet Explorer as their default face considerable uncertainty and confusion in the ordinary course of using Windows 98. [2: ¶171]
>
> Microsoft's refusal to respect the user's choice of default browser fulfilled Brad Chase's 1995 promise to make the use of any browser other than Internet Explorer on Windows "a jolting experience." By increasing the likelihood that using Navigator on Windows 98 would have unpleasant consequences for users, Microsoft further diminished the inclination of OEMs to pre-install Navigator onto Windows. The decision to override the user's selection of non- Microsoft software as the default browser also directly disinclined Windows 98 consumers to use Navigator as their default browser, and it harmed those Windows 98 consumers who nevertheless used Navigator. In particular, Microsoft exposed those using Navigator on Windows 98 to security and privacy risks that are specific to Internet Explorer and to ActiveX controls. [2: ¶172]

Having multiple browsers on the machine would likely confuse users, and thus, Microsoft concluded, many OEMs interested in providing pleasant computing experiences to their users would opt to pre-install only Internet Explorer. However, Microsoft executives felt this step was not strong enough.

> Decision-makers at Microsoft believed that as Internet Explorer caught up with Navigator in quality, OEMs would ultimately conclude that the costs of pre-installing and promoting Navigator, and removing easy access to Internet Explorer, outweighed the benefits. Still, those decision-makers did not believe that Microsoft could afford to wait for the several large OEMs that represented virtually all Windows PCs shipped to come to this desired conclusion on their own. Therefore, in order to bring the behavior of OEMs into line with its strategic goals quickly, Microsoft threatened to terminate the Windows license of any OEM that removed Microsoft's chosen icons and program entries from the Windows desktop or the "Start" menu. It threatened similar punishment for OEMs who added programs that promoted third-party software to the Windows "boot" sequence. These inhibitions soured Microsoft's relations with OEMs and stymied innovation that might have made Windows PC systems more satisfying to users. Microsoft would not have paid this price had it not been convinced that its actions were necessary to ostracize Navigator from the vital OEM distribution channel. [2: ¶203]

OEMs customized the Windows desktops primarily "to make the experience of setting up and learning to use a new PC system easier and less confusing for users, especially novices. By doing so, the OEMs believed, they would increase the value of their systems and minimize both product returns and costly support calls. Since just three calls from a consumer can erase the entire profit that an OEM earned selling a PC system to that consumer, OEMs have an acute interest in making their systems self-explanatory and simple to use." [2: ¶210] Because most computer users

were familiar with Navigator, and because, at the time, Navigator was perceived by nearly all in the industry as superior to Microsoft's Internet Explorer, many of these OEMs included Navigator on their computers. To block this development, Microsoft imposed restrictions on OEMs through its licenses that forbade OEMs from customizing anything, despite the fact that doing so would ultimately provide for a far less satisfying computing experience on the part of the end user.

> The several OEMs that in the aggregate represented over ninety percent of Intel- compatible PC sales believed that the new restrictions would make their PC systems more difficult and more confusing to use, and thus less acceptable to consumers. They also anticipated that the restrictions would increase product returns and support costs and generally lower the value of their machines. Those OEMs that had already spent millions of dollars developing and implementing tutorial and registration programs and/or automatically-loading graphical interfaces in the Windows boot sequence lamented that their investment would, as a result of Microsoft's policy, be largely wasted. Gateway, Hewlett-Packard, and IBM communicated their opposition forcefully and urged Microsoft to lift the restrictions. Emblematic of the reaction among large OEMs was a letter that the manager of research and development at Hewlett- Packard sent to Microsoft in March 1997. He wrote:
>
> > Microsoft's mandated removal of all OEM boot-sequence and auto-start programs for OEM licensed systems has resulted in significant and costly problems for the HP-Pavilion line of retail PC's. Our data (as of 3/10/97) shows a 10% increase in W[indows]95 calls as a % of our total customer support calls . . . . Our registration rate has also dropped from the mid-80% range to the low 60% range.
> >
> > There is also subjective data from several channel partners that our system return rate has increased from the lowest of any OEM (even lower than Apple) to a level comparable to the other Microsoft OEM PC vendors. This is a major concern in that we are taking a step backward in meeting customer satisfaction needs.
> >
> > These three pieces of data confirm that we have been damaged by the edicts that [...] Microsoft issued last fall. . . .
> >
> > From the consumer perspective, we are hurting our industry and our customers. PC's can be frightening and quirky pieces of technology into which they invest a large sum of their money. It is vitally important that the PC suppliers dramatically improve the consumer buying experience, out of box experience as well as the longer term product usability and reliability. The channel feedback as well as our own data shows that we are going in the wrong direction. This causes consumer dissatisfaction in complex telephone support process, needless in-home repair visits and ultimately in product returns. Many times the cause is user misunderstanding of a product that presents too much complexity to the common user. . . .
> >
> > Our Customers hold HP accountable for their dissatisfaction with our products. We bear [...] the cost of returns of our products. We are responsible for the cost of technical support of our customers, including the 33% of calls we get related to the lack of quality or confusion generated by your product. And finally we are responsible for our success or failure in the retail PC market.
> >
> > We must have more ability to decide how our system is presented to our end users.
> >
> > If we had a choice of another supplier, based on your actions in this area, I assure you [that you] would not be our supplier of choice.
> >
> > I strongly urge you to have your executives review these decisions and to change this unacceptable policy.

Despite the high costs that Microsoft's demands imposed on them, the OEMs obeyed the restrictions because they perceived no alternative to licensing Windows for pre-installation on their PCs. [2: ¶¶214–215]

Microsoft engaged in similar tactics with Internet Access Providers to prevent Netscape from success in that channel [2: ¶¶242–310]: The company licensed Internet Explorer and related development kits to hundreds of IAPs for no charge (though those companies were all willing to pay large sums for the software), then entered into exclusivity contracts with the largest IAPs in exchange for valuable promotion within Windows, and even offered to pay back IAPs for any contractual obligations they had with Netscape.

The District Court summarized Microsoft's actions relating to the OEM and IAP channels:

> Neither the desire to bolster demand for Windows, nor the prospect of ancillary revenues, explains the lengths to which Microsoft has gone. For one thing, loading Navigator makes Windows just as Internet-ready as including Internet Explorer does. Therefore, Microsoft's costly efforts to limit the use of Navigator on Windows could not have stemmed from a desire to bolster consumer demand for Windows. Furthermore, there is no conceivable way that Microsoft's costly efforts to induce Apple to pre-install Internet Explorer on Apple's own PC systems could have increased consumer demand for Windows. [2: ¶141]

In other words, the only conceivable reason for Microsoft's actions was to maintain and extend its monopoly.

Microsoft's flagrant bullying of OEMs and IAPs succeeded in closing off distribution channels for Netscape's browser and therefore Sun's Java technology as well. However, Microsoft was not content with this—Microsoft further destroyed any chance of either Netscape's or Java's survival, with or without the OEM and IAP distribution channels. Microsoft did this by ensuring the incompatibility of both software packages with Windows and Windows-based software. Because Windows is the *de facto* standard operating system for the world's desktop computers, this tactic would ensure that only an insignificant portion of the world's computer users would use these two software packages. Therefore, the Windows monopoly would never be challenged by these alternative technologies.

Microsoft changed its Windows API to make existing software incompatible. Microsoft continually modifies the Windows API, and each of these modifications makes software written for Windows immediately out-of-date because that software no longer conforms to the latest specification of the Windows API. This applies to not only applications but hardware and compatible operating systems as well. "IBM discovered this to its dismay in the mid-1990s when it failed, despite a massive investment, to clone a sufficiently large part of the 32-bit Windows APIs. In short, attempting to clone the 32-bit Windows APIs is such an expensive, uncertain undertaking that it fails to present a practical option for a would-be competitor to Windows." [2: ¶52]

The Windows API keeps changing, and software vendors are forced to keep up with the changes because Microsoft ensures that the latest version of the operating system is always being shipped.

> Microsoft takes pains to ensure that the versions of its operating system that OEMs pre-install on new PC systems are the most current. It does this, in part, by increasing the price to OEMs of older versions of Windows when the newer versions are released. [2: ¶57]

Therefore, computer users have no choice but to move onward to the latest version of the operating system, whether it is an improvement over the old version or not. Software vendors have no choice but to update their software to

comply with the new operating system interface, if they expect to sell any software to those computer users who have recently purchased their computers and therefore have the latest version of Windows. It is through this rapid update mechanism that Microsoft keeps consumers and software vendors chained to the latest version of Windows; moreover, through this mechanism Microsoft can harm any software vendor it chooses.

Netscape's browser ran on the pre-Windows95 versions of Windows, but for the Windows95 release of the operating system, Microsoft changed the API so that Netscape's browser would no longer work. This required a re-write of Netscape's browser if Netscape wanted their browser to run on the (then highly anticipated) Windows95 release. The following paragraphs illustrate the power that Microsoft wields over other companies through its ability to modify the Windows API:

> Microsoft knew that Netscape needed certain critical technical information and assistance in order to complete its Windows 95 version of Navigator in time for the retail release of Windows 95. Indeed, Netscape executives had made a point of requesting this information, especially the so-called Remote Network Access ("RNA") API, at the June 21 meeting. As was discussed above, the Microsoft representatives at the meeting had responded that the haste with which Netscape received the desired technical information would depend on whether Netscape entered the so-called "special relationship" with Microsoft. Specifically, Microsoft representative J. Allard had told [Netscape CEO James] Barksdale that the way in which the two companies concluded the meeting would determine whether Netscape received the RNA API immediately or in three months. [2: ¶90]

> Although Netscape declined the special relationship with Microsoft, its executives continued, over the weeks following the June 21 meeting, to plead for the RNA API. Despite Netscape's persistence, Microsoft did not release the API to Netscape until late October, i.e., as Allard had warned, more than three months later. The delay in turn forced Netscape to postpone the release of its Windows 95 browser until substantially after the release of Windows 95 (and Internet Explorer) in August 1995. As a result, Netscape was excluded from most of the holiday selling season. [2: ¶91]

As we have said, because the Windows interface is such a predominant feature in the high-tech industry, any modification to the interface affects potentially every piece of hardware or software that is compatible with Windows. By changing the Windows API, Netscape's software was rendered incompatible with the newest version of Windows and therefore essentially useless.

Microsoft selectively disseminated the details of its Windows API changes. Simply modifying an API is only harmful to the extent that other companies need to expend possibly large resources to play catch-up and to update their products to be compatible with the new specification of the API. As the quotes from the previous section show, the power to modify APIs becomes devastating when those APIs are not public information—i.e. when information regarding those APIs may be handed out selectively. Because Microsoft withheld crucial information from Netscape, Microsoft was able to finish their browser ahead of Netscape, and Netscape was not able to release their Windows95-compatible browser until months after Windows95 was released.

Microsoft behaved in similar ways in most of its other dealings with Netscape.

> Microsoft similarly withheld a scripting tool that Netscape needed to make its browser compatible with certain dial-up ISPs. Microsoft had licensed the tool freely to ISPs [Internet Service Providers] that wanted it, and in fact had cooperated with Netscape in drafting a license agreement that, by mid-July 1996, needed only to be signed by an authorized Microsoft executive to go into effect. There the process halted, however. In mid-August, a Microsoft representative informed Netscape that senior executives at Microsoft had decided to link the grant of the license to the resolution of all open issues between the companies. Netscape never received a license to the scripting tool, and as a result, was unable to do business with certain ISPs for a time. [2: ¶92]

It is well known that Microsoft implements numerous "undocumented" features in its operating system, usually to the benefit of its own software. "Microsoft has special knowledge of its own products, and it alone chooses which functionalities in its products are to be documented and which are to be left undocumented." [2: ¶179] Furthermore, Microsoft frequently enters into agreements with some software developers in which those developers are granted "preferred" status and are therefore given access to more (but not necessarily all) information about the undocumented features of the operating system [2: ¶ 84].

Because software running on a PC cannot use any hardware feature without the consent of the operating system, restricting access to the operating system cuts off the ability of software to do anything useful. Microsoft's dealings with Netscape are simply one example that demonstrate the enormous power that Microsoft wields and the extent to which the company has used that power to stifle individual corporations and to reward others.

Microsoft subverted the Java interface standard. As mentioned, Java is a technology that enables computer programmers to write programs that will run on any operating system, and any hardware, anywhere. The fundamental concept behind Java is the idea that such interoperability is far more valuable than the ability to execute programs quickly, because the performance of computer hardware is improving at an astounding rate, and this will more than make up for any lack of performance seen in Java. Therefore, developers who write Java code are typically those who are willing to give up a small amount of performance in return for a larger market in which to sell their software.

However, this is not what Microsoft wants—Microsoft is best served if all software that is compatible with Windows is completely dependent on Windows:

> For Microsoft, a key to maintaining and reinforcing [the difficulty of potential competitors to create an alternative to Windows] has been preserving the difficulty of porting applications from Windows to other platforms [e.g. operating systems], and vice versa. In 1996, senior executives at Microsoft became aware that the number of developers writing network-centric applications in the Java programming language had become significant, and that Java was likely to increase in popularity among developers. Microsoft therefore became interested in maximizing the difficulty with which applications written in Java could be ported from Windows to other platforms, and vice versa. [2: ¶386]

Microsoft set out not to restrict Sun's ability to sell Java technology, but to extinguish the entire Java phenomenon. Microsoft recognized that the best way to do this would be to ensure that a significant portion of developers wrote Java code that would not, in fact, run on any operating system, any hardware, anywhere—Microsoft ensured that Java

written to be compatible with Windows ran correctly only on Windows, and not on any other operating system. The District Court investigated and analyzed Microsoft's campaign against Java:

> Specifically, the District Court found that Microsoft took four steps to exclude Java from developing as a viable cross-platform threat: (a) designing a JVM incompatible with the one developed by Sun; (b) entering into contracts, the so-called "First Wave Agreements," requiring major ISVs to promote Microsoft's JVM exclusively; (c) deceiving Java developers about the Windows-specific nature of the tools it distributed to them; and (d) coercing Intel to stop aiding Sun in improving the Java technologies. [1: p. 52]

"JVM" stands for Java Virtual Machine and is the primary component that is required to be present on a computer if that computer is to run Java programs. By creating a JVM that was incompatible with Sun's and ensuring that a substantial fraction of developers used it and not Sun's, Microsoft went far beyond denying Sun a market in which to sell its own JVM—Microsoft ensured that Java itself would fail.

> On March 12, 1996, Sun signed an agreement granting Microsoft the right to distribute and make certain modifications to Sun's Java technologies. Microsoft used this license to create its own Java development tools and its own Windows-compatible Java runtime environment. Because the motivation behind the Sun-sponsored effort ran counter to Microsoft's interest in preserving the difficulty of porting, Microsoft independently developed methods for enabling "calls" to "native" Windows code that made porting more difficult than the method that Sun was striving to make standard. Microsoft implemented these different methods in its developer tools and in its JVM. Microsoft also discouraged its business allies from aiding Sun's effort. For example, Gates told Intel's CEO in June 1996 that he did not want the Intel Architecture Labs cooperating with Sun to develop methods for calling upon multimedia interfaces in Windows. [2: ¶388]

> Sun had already developed a JVM for the Windows operating system when Microsoft began work on its version. The JVM developed by Microsoft allows Java applications to run faster on Windows than does Sun's JVM, Findings of Fact p 389, but a Java application designed to work with Microsoft's JVM does not work with Sun's JVM and vice versa. Id. p 390. The District Court found that Microsoft "made a large investment of engineering resources to develop a high-performance Windows JVM," id. p 396, and, "[b]y bundling its … JVM with every copy of [IE] … Microsoft endowed its Java runtime environment with the unique attribute of guaranteed, enduring ubiquity across the enormous Windows installed base," id. p 397. As explained above, however, a monopolist does not violate the antitrust laws simply by developing a product that is incompatible with those of its rivals. … In order to violate the antitrust laws, the incompatible product must have an anticompetitive effect that outweighs any procompetitive justification for the design. [1: pp. 51–52]

The Court of Appeals found that simply creating this non-standard JVM was not in violation of antitrust laws. However, the Court also stated that "to violate the antitrust laws, the incompatible product must have an anticompetitive effect that outweighs any procompetitive justification for the design." It is our position that this is exactly the case here. The fundamental point of Java is to promote cross-platform compatibility. Any computer engineer knows that one can sacrifice generality for better performance; such trade-offs are trivial to make, and it is little wonder that Microsoft was able to create a high-performance JVM by sacrificing compatibility. However, doing so outweighs the procompetitive justification for the design, because, as we have said, the fundamental purpose of Java is not to achieve the highest possible performance but to provide generality—i.e. compatibility with all operating systems and all hardware platforms.

Had Java developers known what Microsoft was doing, it is quite possible that they would have opted for Sun's JVM implementation over Microsoft's. In addition, Intel had built a high-performance JVM that did comply with the Java standard, and it is likely that the Java developers would have chosen this over either Microsoft's implementation or Sun's implementation. However, Microsoft prevented either from happening.

First, Microsoft entered into agreements with numerous independent software vendors (ISVs) in which it forbade the ISVs from using any Java technology offered by Sun.

> Recognizing ISVs as a channel through which Java runtime environments that complied with Sun's standards could find their way onto Windows PC systems, Microsoft induced ISVs to distribute Microsoft's version instead of a Sun-compliant one. First, Microsoft made its JVM available to ISVs separately from Internet Explorer so that those uninterested in bundling browsing software could nevertheless bundle Microsoft's JVM. Microsoft's David Cole revealed the motivation for this step in a message he wrote to Jim Allchin in July 1997: "[W]e've agreed that we must allow ISVs to redistribute the Java VM standalone, without IE. ISVs that do this are bound into Windows because that's the only place the VM works, and it keeps them away from Sun's APIs." [2: ¶400]

> Microsoft took the further step of offering valuable things to ISVs that agreed to use Microsoft's Java implementation. Specifically, in the First Wave agreements that it signed with dozens of ISVs in 1997 and 1998, Microsoft conditioned early Windows 98 and Windows NT betas, other technical information, and the right to use certain Microsoft seals of approval on the agreement of those ISVs to use Microsoft's version of the Windows JVM as the "default." Microsoft and the ISVs all read this requirement to obligate the ISVs to ensure that their Java applications were compatible with Microsoft's version of the Windows JVM. The only effective way to ensure compatibility with Microsoft's JVM was to use Microsoft's Java developer tools, which in turn meant using Microsoft's methods for making native calls and (unless the developers were especially wary and sophisticated) Microsoft's other Java extensions. Thus, a very large percentage of the Java applications that the First Wave ISVs wrote would run only on Microsoft's version of the Windows JVM. […] The record contains no evidence that the relevant provision in the First Wave agreements had any purpose other than to maximize the difficulty of porting Java applications between Windows and other platforms. [2: ¶401]

Nonetheless, developers, especially those "wary and sophisticated," might have chosen to use JVMs written by a party other than Microsoft. However, Microsoft assuaged potential fears of developers by essentially committing fraud: The company deceived developers as to the portability of code written for its JVM.

> Microsoft's "Java implementation" included, in addition to a JVM, a set of software development tools it created to assist ISVs in designing Java applications. The District Court found that, not only were these tools incompatible with Sun's cross-platform aspirations for Java--no violation, to be sure--but Microsoft deceived Java developers regarding the Windows-specific nature of the tools. Microsoft's tools included "certain 'keywords' and 'compiler directives' that could only be executed properly by Microsoft's version of the Java runtime environment for Windows." Id. p 394; see also Direct Testimony of James Gosling p 58, reprinted in 21 J.A. at 13959 (Microsoft added "programming instructions … that alter the behavior of the code."). As a result, even Java "developers who were opting for portability over performance … unwittingly [wrote] Java applications that [ran] only on Windows." Conclusions of Law, at 43. That is, developers who relied upon Microsoft's public commitment to cooperate with Sun and who used Microsoft's tools to develop what Microsoft led them to believe were cross-platform applications ended up producing applications that would run only on the Windows operating system.

> When specifically accused by a PC Week reporter of fragmenting Java standards so as to prevent cross-platform uses, Microsoft denied the accusation and indicated it was only "adding rich platform support" to what remained a cross-platform implementation. An e-mail message internal to Microsoft, written shortly after the conversation with the reporter, shows otherwise:
>
> > [O]k, i just did a followup call…. [The reporter] liked that i kept pointing customers to w3c standards [(commonly observed internet protocols)]…. [but] he accused us of being schizo with this vs. our java approach, i said he misunderstood [--] that [with Java] we are merely trying to add rich platform support to an interop layer…. this plays well…. at this point its [sic] not good to create MORE noise around our win32 java classes. instead we should just quietly grow j++ [(Microsoft's development tools)] share and assume that people will take more advantage of our classes without ever realizing they are building win32-only java apps.
>
> GX 1332, reprinted in 22 J.A. at 14922-23.
>
> Finally, other Microsoft documents confirm that Microsoft intended to deceive Java developers, and predicted that the effect of its actions would be to generate Windows-dependent Java applications that their developers believed would be cross-platform; these documents also indicate that Microsoft's ultimate objective was to thwart Java's threat to Microsoft's monopoly in the market for operating systems. One Microsoft document, for example, states as a strategic goal: "Kill cross-platform Java by grow[ing] the polluted Java market." GX 259, reprinted in 22 J.A. at 14514; see also id. ("Cross-platform capability is by far the number one reason for choosing/using Java.") (emphasis in original). [1: pp. 55–56]

It is interesting to note, in the last excerpt, that even Microsoft officials recognized that Java's strength lay in its cross-platform interoperability.

Lastly, Microsoft killed Intel's high-performance JVM, which was fully compliant with the Java standard and could therefore easily have overtaken Microsoft's JVM.

> The District Court held that Microsoft also acted unlawfully with respect to Java by using its "monopoly power to prevent firms such as Intel from aiding in the creation of cross-platform interfaces." Conclusions of Law, at 43. In 1995 Intel was in the process of developing a high-performance, Windows-compatible JVM. Microsoft wanted Intel to abandon that effort because a fast, cross-platform JVM would threaten Microsoft's monopoly in the operating system market. At an August 1995 meeting, Microsoft's Gates told Intel that its "cooperation with Sun and Netscape to develop a Java runtime environment … was one of the issues threatening to undermine cooperation between Intel and Microsoft." Findings of Fact p 396. Three months later, "Microsoft's Paul Maritz told a senior Intel executive that Intel's [adaptation of its multimedia software to comply with] Sun's Java standards was as inimical to Microsoft as Microsoft's support for non-Intel microprocessors would be to Intel." Id. p 405.
>
> Intel nonetheless continued to undertake initiatives related to Java. By 1996 "Intel had developed a JVM designed to run well … while complying with Sun's cross-platform standards." Id. p 396. In April of that year, Microsoft again urged Intel not to help Sun by distributing Intel's fast, Sun-compliant JVM. Id. And Microsoft threatened Intel that if it did not stop aiding Sun on the multimedia front, then Microsoft would refuse to distribute Intel technologies bundled with Windows. Id. p 404.
>
> Intel finally capitulated in 1997. [1: pp. 56–57]

The downfall of Java is important because, at the time, Java represented a potential competitor to the Windows hegemony. Currently, most software developers write code for the Windows platform because it represents the largest

possible market for their software. As we have mentioned before, code written for one operating system does not run on another without considerable effort in re-writing that code so that, instead of using the first operating system's interface, it uses the second's interface. This rewriting of a software program is called "porting" the code and represents a substantial undertaking. Java offered developers a way to increase the size of the market for their software, by allowing developers to write software that would run not only on Windows but on every operating system and hardware platform imaginable. As the Court's records show, in the mid-1990's developers recognized this benefit and were moving in large numbers to use Java [2: ¶386]. However, the high-tech industry is very fickle, a by-product of its highly dynamic nature, and if a technology fails to catch on the first time around, it has a difficult road ahead of it. By taking away the main benefit of Java—its promise of universal compatibility—Microsoft allowed developers to become cool to the idea, and this essentially killed it.

Had the Java vision been realized, it would have offered competition in the PC operating systems market. Microsoft waged its campaign against Java for exactly this reason: Had the Java vision come true, it would have weakened the stronghold of Windows, because a user's choice of underlying operating system would have become less important. This would have increased the likelihood for computer users to consider alternative or "fringe" operating systems such as Unix, Linux, BeOS, Mac OS, etc. Microsoft's successful subversion of Java took much of the momentum out of the Java movement and in so doing ensured that neither Java nor Java-enabled operating systems would present any significant competition with Windows in the PC operating systems market.

Microsoft attempted to subvert the HTML interface standard. Microsoft attempted a similar destruction or undermining of the HTML standard (the language of the world-wide web), by creating its own incompatible "extensions" to the language and forcing as many developers as possible to use those extensions. As early as 1995, Microsoft had developed HTML control words that were incompatible with all browsers but its own Internet Explorer. For computer users viewing the web through any browser but Microsoft's, web pages using these control words would either display incorrectly or fail to display at all. Evidently, Microsoft hoped that, if it could make Internet browsing a confusing and irritating experience for enough people (those not using Internet Explorer on Windows), the popularity of the Internet would wane, and the "inflection point" that the Internet represented would not have the potential to injure Microsoft.

Microsoft entered into numerous agreements to spread its non-standard HTML. For example, in 1997, the company set up agreements with website developers (also called Internet Content Providers, or ICPs):

> [T]he agreements required the ICPs, in designing their Web sites, to employ certain Microsoft technologies such as Dynamic HTML and ActiveX. Some of the agreements actually required the ICPs to create "differentiated content" that was either available only to Internet Explorer users or would be more attractive when viewed with Internet Explorer than with any "Other Browser." For example, the agreement with Intuit provided: "Some differentiated content may be available only to IE users, some may simply be 'best when used with IE,' with acceptable degradation when used with other browsers." [2: ¶322]

In addition, Microsoft began forcing even OEMs, such as Compaq, to put non-standard HTML features into their websites.

> When Compaq eventually [capitulated to Microsoft's wishes], it did so because its senior executives had decided that the firm needed to do what[ever] was necessary to restore its special relationship with Microsoft. […] Compaq agreed to offer Internet Explorer as the preferred browser product for its Internet products and to use two or more of Microsoft's hypertext markup language ("HTML") extensions in the home page for each of those products. [2: ¶233]

Microsoft targeted software developers as well:

> Microsoft also targeted individual ISVs [Independent Software Vendors] directly, extracting from them commitments to make their Web-centric applications reliant on technology specific to Internet Explorer. [2: ¶337]

> In dozens of "First Wave" agreements signed between the fall of 1997 and the spring of 1998, Microsoft has promised to give preferential support […] to important ISVs that agree to certain conditions. One of these conditions is that the ISVs use Internet Explorer as the default browsing software for any software they develop with a hypertext-based user interface. Another condition is that the ISVs use Microsoft's "HTML Help," which is accessible only with Internet Explorer, to implement their applications' help systems. [2: ¶339]

> By exchanging its vital support for the agreement of leading ISVs to make Internet Explorer the default browsing software on which their products rely, Microsoft has ensured that many of the most popular Web-centric applications will rely on browsing technologies found only in Windows. [2: ¶340]

Despite Microsoft's efforts, consumer interest in the Internet did not wane, and irritation at the inability to view certain web pages did not drive users to forgo browsing the web. It is likely that Microsoft's failure in this particular campaign was due to the fact that Microsoft attempted to subvert the standard only once it was already in wide use—i.e. once it had already succeeded. In comparison, Java was killed in its infancy, before it had reached a critical mass of support.

—

It is important to note that these are not isolated incidents chosen simply to disparage Microsoft. The record shows that Microsoft saw similar threats from other quarters and responded to them in much the same manner as the maneuvers described above. The following are three examples of how Microsoft responded to quash the development of either direct competitors to Windows or "middleware" for Windows, as well as a brief look at Microsoft's latest anticompetitive action. Middleware is a type of software program that serves to hide the particulars of the underlying operating system, Java being one example of a middleware system. The very nature of middleware makes it a threat to Microsoft's operating system enterprise because it has the potential to make Windows obsolete:

> A middleware product written for Windows could take over some or all of Windows's valuable platform functions—that is, developers might begin to reply upon APIs exposed by the middleware for basic routines rather than relying upon the API set included in Windows. If middleware were written for multiple operating

> systems, its impact could be even greater. The more developers could rely upon APIs exposed by such middleware, the less expensive porting to different operating systems would be. Ultimately, if developers could write applications relying exclusively on APIs exposed by middleware, their applications would run on any operating system on which the middleware was also present. [1: p. 18]

In the early 1990's, Intel created multimedia features in their hardware that would enable computers to play movies and high-resolution computer games, but Microsoft refused to make these hardware features available through its operating system. Typically, as shown in Figure 1, hardware devices and other sophisticated hardware features are available to software only through the operating system, and if the operating system fails to support the hardware, no software running on that system can access that hardware. Intel began developing software called Native Signal Processing that would allow programmers to use the new multimedia features without having to go through Windows, so as to encourage software developers to write multimedia programs. In response, Microsoft threatened to terminate compatibility with Intel's chips if Intel persisted in offering programmers a way to run software on its hardware without having to use Windows. Intel was ultimately forced by Microsoft into giving up its software experiments. Five years after the fact, Microsoft finally integrated into Windows most, but not all, of the multimedia capabilities that Intel had developed. The Court's explanation for Microsoft's behavior was a "fear at Microsoft that the NSP software would render ISVs, device manufacturers, and (ultimately) consumers less dependent on Windows. Without this fear, Microsoft would not have subjected Intel to the level of pressure that it brought to bear in the summer of 1995." [2: ¶¶94–103]

IBM expended enormous effort to reverse-engineer the Windows API and thereby build their own, fully interchangeable, version of the Windows operating system. IBM's operating system was called OS/2 Warp, and, because IBM faithfully reproduced a substantial portion of the Windows API, OS/2 Warp was able to run Windows applications directly. The operating system was billed with the advertising slogan, "A Better Windows Than Windows," which by all accounts was true: The operating system was faster than Windows, it was more reliable than Windows (it did not crash nearly as easily), and it ran most Windows applications flawlessly. Because it was faster and more stable than Microsoft's operating system, it had the potential to weaken the appeal of Windows. In response, Microsoft swiftly changed the Windows APIs, leaving OS/2 Warp offering an obsolete version of the operating system interface. IBM could not keep pace with the interface changes and ultimately gave up on OS/2 Warp as a PC operating system. [2: ¶52, ¶¶115–132]

Apple created its QuickTime standard to be a multimedia authoring tool and audio/video publishing format that would run on any computer, whether Macintosh-based or Windows-based. Microsoft saw this as a threat because QuickTime enabled software developers to write multimedia programs that would run on Windows but that did not depend directly on Windows. Microsoft developed its own multimedia standard (DirectX) and threatened to develop multimedia content-development software that was incompatible with Apple's and to expend every resource available to ensure that it won out over QuickTime, unless Apple ensured that QuickTime would not run equally well on all platforms. Microsoft's proposal was for Apple to make their Windows-based QuickTime software entirely dependent on Windows. "[Apple's CEO Steve] Jobs reserved comment during the meeting with the Microsoft representatives,

but he explicitly rejected Microsoft's proposal a few weeks later. Had Apple accepted Microsoft's proposal, Microsoft would have succeeded in limiting substantially the cross-platform development of multimedia content." [2: ¶¶106–109]

Windows XP is Microsoft's latest version of the Windows operating system, released in November 2001in the midst of the trial's conclusion. The operating system flouts the very same laws that Microsoft was charged with violating in the first place. The operating system disables the software of competitors—including Apple's QuickTime media player, AOL Time Warner's America On-Line version 6.0, and RealJukebox from RealNetworks [23]—and supplants them with Microsoft's proprietary alternatives. Note that these are the very companies with which Microsoft had similar run-ins previously [1,2]. Not only is the browser tied to the operating system, but so is nearly every aspect of the computer user's experience, from Internet content to popular multimedia formats. The writing on the wall is very clear: No software vendor is safe—Microsoft can destroy any competitor's software it chooses and replace it with its own.

Solutions to the Problem

According to the Supreme Court, there are four primary goals for an antitrust remedy. "The Supreme Court has explained that a remedies decree in an antitrust case should seek to 'unfetter a market from anticompetitive conduct,' … to 'terminate the illegal monopoly, deny to the defendant the fruits of its statutory violation, and ensure that there remain no practices likely to result in monopolization in the future.'" [1: pp. 99-100] We propose a multi-part remedy that addresses each of these goals.

Most Importantly: Prevent Future Monopolization

We treat the last goal first because we believe it to be the most important. The first three goals—to restore competition, to terminate the illegal monopoly, and to deny the illegal monopoly the fruits of its misconduct—are addressed in the next section.

As the Supreme Court indicates, an antitrust remedy is not sufficient if it merely returns us to the status quo [9,10]. An appropriate remedy must also seek to prevent future monopolization.

We believe that the best way to prevent future misconduct from Microsoft is to look at those of its actions that are problematic and devise a remedy that prevents similar maneuvers in the future. The list, again, of Microsoft's actions:

1. Microsoft created an alternative browser to Netscape's.

Action #1 alone is an example of behavior that is perfectly legal and should remain so: A monopoly clearly should not be prevented from competing in a market just because of its status as a monopoly. The rest of the actions, however, are objectionable:

2. Microsoft brought pressure to bear on OEM and IAP channels.

3. Microsoft changed its Windows API to make existing software incompatible.

4. Microsoft selectively disseminated the details of its Windows API changes.

5. Microsoft subverted the Java interface standard.

6. Microsoft attempted to subvert the HTML interface standard.

It is this second group of tactical maneuvers, numbers 2 through 6, that we will explore for direction, and doing so leads us to further remedial steps to prevent future misconduct. Action #2, pressuring OEMs to pre-load only Internet Explorer, and action #4, selectively disseminating information on its changes to the Windows API, are both obvious violations of the Sherman Act. For instance, as detailed earlier, Microsoft's manipulation of the Windows API harmed both IBM and Netscape, and therefore action #2 is part of the illegality. The other actions represent very serious and very effective abuses of power, and Microsoft violated the Sherman Act when the company used its abilities to harm other companies through these actions. For instance, Microsoft's undermining of the Java standard harmed Sun, and therefore action #5 is part of the illegality. Clearly, it is imperative to prevent all of the listed behaviors if we are to prevent future monopolistic abuses by Microsoft.

Microsoft undertook these anticompetitive actions because it believed its Windows API—its operating systems interface—to be threatened by other technologies that could potentially supplant it. As we have said, like contracts, interfaces are extremely powerful things. Compared to a tangible product, an internal interface differs in its impact on economic systems because not only are end-consumers dependent on it, but other hardware and software vendors that build products compatible with the interface are dependent on it as well. Furthermore, the effect of modifying an interface on which competing companies depend is markedly different from the effect of simply developing a competing product: Whereas a new end-product simply offers consumers an alternative to the competitor's product that will have to stand the test of the marketplace, modifying an interface can make the competing company's products instantly obsolete—because as soon as the modification is made the other company's product no longer conforms to the interface, and an interface is ultimately more influential in its scope than any particular product that embodies it. Therefore it is clear that a monopoly controlling a widely used interface is perfectly capable of controlling its direct competitors, its indirect competitors, and even those corporations that provide complementary products and do not compete with it in any way.

The obvious conclusion is that monopolies should not be allowed to control widely used interfaces. We propose two remedies that would prevent future misconduct:

A. Microsoft's widely used interfaces (in particular the Windows API) should be turned over to the public—i.e. Microsoft should give up its right to modify at will those of its interfaces that have become *de facto* standards. The authors would prefer to see the interfaces handed over to the IEEE (the Institute of Electrical and Electronics Engineers), as the IEEE is an independent, non-profit body already involved in the

specification of many public computer interface standards, both hardware and software. However, the choice of a governing body is a matter of opinion.

B. Microsoft should adhere to existing *de facto* standard interfaces: The company should not develop or fund the development of any interface that provides substantially similar function as an existing standard but that is incompatible with that standard.

Note that remedial step A differs from Judge Jackson's interim provision in which Microsoft was compelled to provide all information about the Windows API to all developers in a timely manner and equally to all [13]. His Final Judgment stated that, until full implementation of the divestiture plan which he imposed, among other things the following was to be required of Microsoft:

> b. Disclosure of APIs, Communications Interfaces and Technical Information. Microsoft shall disclose to ISVs, IHVs, and OEMs in a Timely Manner, in whatever media Microsoft disseminates such information to its own personnel, all APIs, Technical Information and Communications Interfaces that Microsoft employs to enable -
>
> i. Microsoft applications to interoperate with Microsoft Platform Software installed on the same Personal Computer, or
> ii. a Microsoft Middleware Product to interoperate with Windows Operating System software (or Middleware distributed with such Operating System) installed on the same Personal Computer, or
> iii. any Microsoft software installed on one computer (including but not limited to server Operating Systems and operating systems for handheld devices) to interoperate with a Windows Operating System (or Middleware distributed with such Operating System) installed on a Personal Computer.
>
> To facilitate compliance, and monitoring of compliance, with the foregoing, Microsoft shall create a secure facility where qualified representatives of OEMs, ISVs, and IHVs shall be permitted to study, interrogate and interact with relevant and necessary portions of the source code and any related documentation of Microsoft Platform Software for the sole purpose of enabling their products to interoperate effectively with Microsoft Platform Software … [13]

Judge Jackson's judgment essentially makes the previously "undocumented" features of Windows openly available to all[2], and the judgment prevents Microsoft from withholding key information as it did to delay Netscape's offering of a Windows95-compatible browser. Our proposed remedy A would push Judge Jackson's judgment a step further—our proposed remedy would take the Windows API from Microsoft's control and hand it over to an independent body as a public interface. Note that this does *not* mean that Microsoft would have to divulge any source code to the public; only the Windows API would become public.

Public interfaces, often called public standards, are those that are not proprietary, are controlled by an independent body (such as ANSI, the American National Standards Institute; IEEE, the Institute of Electrical and Electronics

---

2. The "undocumented" features of Windows are those features that exist in the operating system but that are unavailable to all but a select group by virtue of the fact that the features are not specified in any publicly available document.

Engineers; IETF, the Internet Engineering Task Force; etc.), and are typically designed to make computers of different types compatible with each other. Generally, all who wish to participate in the specification of a public standard are allowed to contribute to the effort: Because of the public nature of such discussions, no one is prevented from playing a part. Conversely, no individual or corporation is allowed to make unilateral decisions about the standard. A perfect example of a public interface is the World-Wide Web: It uses the Internet Protocol—a public standard—to transmit documents written in HTML—also a public standard—from web servers to web browsers, using yet another public standard called HTTP (hypertext transmission protocol) that runs on top of the Internet Protocol. Any computer anywhere that supports these standards can browse the web, whether it is Windows-based or PalmOS-based or Macintosh-based or Unix-based or BeOS-based or Linux-based, and the fact that the standards are public increases the chance that any particular operating system will support them. The fact that the web uses nothing but public standards is one of the characteristics that attracted developers to the web in the first place: Any programmer can build a web server or web browser and know that it will work with the web; moreover, because the standards are public, the programmer need not pay any royalties to anyone; lastly, because the standards cannot be changed by any one person or company acting alone, the programmer can be assured that his work will not suddenly become obsolete due to unannounced changes to the various interfaces.

A monopoly should certainly be allowed to help define these public computing interfaces—who better to help realize the future of computing than a company with years of experience in the computer industry? However, a company should not be allowed to use its monopoly status to do an end-run around an existing public standard and undermine the efforts of others to make computers more compatible with each other, more useful, and easier to use.

These two proposed remedial steps together would prohibit Microsoft from doing just this: Together, they would ensure that Microsoft will not be able to create, either directly or indirectly (i.e. by persuading smaller companies to do its bidding), new products that undermine public standards. Moreover, Microsoft's own primary interface, the Windows API, would become a public standard, which would prevent Microsoft from modifying that interface at will, as the company has done in the past to ensure that Netscape's browser would fail to work with Windows95 and to ensure that IBM would not be able to keep up with its reverse-engineering effort that made OS/2 possible.

Unfetter the Market from Anticompetitive Conduct, Terminate the Illegal Monopoly, and Deny Microsoft the Fruits of its Misconduct

As mentioned in the previous section, the court, in fashioning remedies, should go beyond merely restoring the status quo. The court should impose a remedy that will cure the effects of the illegal conduct and, at the same time, prevent the continuation of the misconduct and prevent the violator from reaping benefits in the future from their conduct [9, 10].

Because the "relevant market" for a monopoly is defined as including all products that are "reasonably interchangeable by consumers for the same purposes" [1: p. 15], a monopoly exists by definition if there are no such

products available. No product currently exists that is reasonably interchangeable with the Windows operating system, and Microsoft's present monopoly will not be terminated until one does exist. The solution is rather clear: Microsoft eliminated the threat of potential competitors to its Windows operating system by exclusionary tactics, and therefore it should bear the responsibility of funding the production of such a competitor. Or perhaps more than one. Doing so would introduce competition into the market, would end Microsoft's monopoly, and would deny Microsoft the fruits of its anticompetitive behavior.

We propose the following additional remedial steps, in addition to those already discussed:

C. All non-compliant Java technology developed by Microsoft (its Java Virtual Machine and its development environment) should be destroyed, and Microsoft should be prevented from distributing it in the future, either directly or indirectly.

D. Microsoft should pay Intel to finish the development of its high-performance Java Virtual Machine, and Microsoft should also pay at least the initial costs of marketing the product.

E. The non-compliant Java Virtual Machine in Microsoft's browser should be removed and replaced with Intel's compliant and high-performance JVM.

F. Microsoft should pay Sun Microsystems to finish development of Java technologies and market the product. In particular, the marketing of Java technologies should be handled in much the same way as Microsoft's "evangelizing" of the Windows platform (Microsoft spends hundreds of millions of dollars per year inducing independent software developers to write applications for Windows [2: ¶43]). Microsoft should pay Sun a large one-time sum, perhaps equal to a year's worth of Windows evangelizing, to be used by Sun to seed development of Java applications by independent software vendors.

G. Microsoft should pay IBM to update OS/2 to be compliant with the latest version of the Windows API (e.g. Windows XP), so as to provide an alternative operating system to Windows. Because Microsoft's ability to manipulate the Windows API at will is what drove OS/2 out of the PC market last time, such manipulation should be regulated in the future for OS/2 to remain a reasonably interchangeable alternative. Thus, remedial steps A and B are necessary.

H. Microsoft should make its browser and MSN portal compatible with the World-Wide Web Consortium's HTML, HTTP, and XML interface standards. In particular, the MSN portal should not shut out access by browsers other than Internet Explorer, as it presently does [25].

These remedies would create two alternatives to the Windows operating system. The first alternative is based on Java middleware, which could be disseminated by itself or bundled with a browser. As the appellate court stated in its opinion, this is one of the ways in which to make the PC operating-systems market competitive:

> If a consumer could have access to the applications he desired—regardless of the operating system he uses—simply by installing a particular browser on his computer, then he would no longer feel compelled to select Windows in order to have access to those applications; he could select an operating system other than Windows based solely upon its quality and price. In other words, the market for operating systems would be competitive. [1: p. 29]

Remedies C–F provide for the Java alternative. This would create a layer of middleware that is truly cross-platform. If it is marketed well, the average end-consumer might become familiar with it, thereby making it a viable alternative to Windows. If Microsoft were to be prevented from undermining the Java standard, software developers could write applications and know that their software would run on any operating system equally well, from Windows to Mac OS to Unix to Linux to BeOS, etc. This would create larger markets for the developers' software (as opposed to writing software that runs on only one operating system) and could also raise consumer interest level in alternative operating systems—provided that enough developers begin to write compelling Java applications, such as a reasonably interchangeable alternative to Microsoft's Office suite. Fundamental to the scheme is a truly cross-platform, high-performance environment, which is what Intel was attempting to help Sun deliver. Therefore, it is crucial that Intel's high-performance JVM take the place of Microsoft's incompatible JVM, and all of Microsoft's non-compliant Java technology be kept off the market and thus out of circulation and unavailable for public consumption.

Because the future widespread availability of Java applications that rival their Windows counterparts in features and sophistication is merely speculative, even with seed funding from Microsoft, it is best to provide a second alternative as well: a full-blown operating system having the same features as Windows and able to run Windows applications. Remedy G provides for the OS/2 alternative: a truly interchangeable alternative to Windows. This was the promise of OS/2 before Microsoft changed the Windows API more rapidly than IBM could reverse-engineer it. With Microsoft funding its development, IBM should be able to bring OS/2 up to the WindowsXP API in a very short time (in software development, emulating an existing product is much simpler than creating something new from scratch).

Remedy H simply ensures that Microsoft would be required to maintain open standards, rather than undermining them.

To avoid simply returning to the status quo, Microsoft must be prevented from performing future acts that mirror its previous acts. If Java compatibility is ensured by the eradication of the non-compliant technology, Microsoft should be prevented from building future incompatible Java technology, else history will be repeated. If there is a revival of OS/2, Microsoft should not be allowed to manipulate the Windows API so as to make the new version of OS/2 instantly obsolete, as it did in the mid-1990's to eliminate the threat of OS/2 Warp. Remedial steps A and B together prevent future monopolistic abuses of this type.

Discussion

Making the Windows API a public standard owned by an independent organization separate from Microsoft and preventing Microsoft from manipulating the interface as it sees fit are both essential if the remedial steps in the previous section are to work for any length of time. If Microsoft has free reign to modify the Windows API, it could

very quickly make Intel's JVM non-compatible with Windows, and, if OS/2 were to become again an alternative to the Windows operating system, Microsoft could quickly make OS/2 obsolete again simply by changing the Windows API. If Microsoft retains control of the Windows API, it can target any company's software and make it instantly obsolete by appropriately modifying the API. By the time the company re-writes its software to reflect the changes, Microsoft could modify the Windows API even further, and thereby ensure that the targeted company would never have a software product that is compatible with the latest version of Windows.

For example, Microsoft's Passport system is a software package bundled with WindowsXP that handles the financial side of e-commerce transactions, and it provides Microsoft with a small fee charged to the user for every e-commerce transaction Passport handles. If e-commerce is to become as pervasive as predicted, Passport represents future revenues that dwarf the revenues Microsoft currently receives from Windows and Office combined: Because Windows-based PCs represent 80–90% of the world's desktop computers, one can expect that 80–90% of the world's e-commerce transactions will be handled through Passport. If a competitor to Passport arises, Microsoft can simply change the Windows API in a matter of minutes, release the new version of the operating system to OEMs, and in so doing make the new e-commerce software incompatible with Windows—which will temporarily bar that competitor from the Windows desktop. It will take time for the new company to build and release a new version of their software that is compatible with the newer version of Windows, and during this time Microsoft will be able to make further modifications to the Windows API that will render the competitor's revision out-of-date. By allowing Microsoft to own the Windows API, we give Microsoft the ability to harm any targeted company indefinitely. Microsoft has already demonstrated that it is willing to use this power: Remember that Netscape's browser ran perfectly well under Windows 3.1, but Microsoft's changes to the API made Netscape's browser incompatible with Windows95. Remember also that Microsoft's latest version of Windows, WindowsXP, makes numerous software packages by its competitors incompatible. There is every reason to believe that Microsoft will continue to use this ability in the future to harm potential competitors in any promising market, and it is therefore critical to any remedies decree that this ability, to modify the Windows API without the permission of its constituents (i.e. independent software and hardware vendors as well as the public consumer), be stripped from Microsoft.

An argument against this remedial step is that it could stifle competition by preventing Microsoft from creating new features at will—by curbing Microsoft's ability to "innovate." In rare cases, new features do require changes to interfaces, but the vast majority of new features tend to work quite well with the old interfaces. For instance, assuming the company is not prohibited from doing so, Microsoft could easily integrate its browser into the Windows operating system without having to change the operating system interface at all. The Unix file I/O interface is an extremely potent example of an interface that has not changed in the decades since its inception yet is still allowing innovation to this day. The interface was defined to allow software to open, read, write, and close files stored on disks. Numerous features have been added to the Unix operating system using the same interface, and—just to name a few examples—software programs can now send messages on the Internet, interact with other programs, use input devices such as mice and drawing tablets, and even talk to smart cards using the same, unchanged, interface as

reading and writing files on disks. Clearly, innovation is not at all stifled by restricting changes to interfaces, while this entire article illustrates numerous potent examples of stifling innovation by *allowing* such changes. Changing the nature of Microsoft's competition with other companies would not hinder its ability to innovate; it would simply force Microsoft to pay more attention to inter-operability, ease-of-use, performance, and reliability, as these would be the characteristics on which their products would be judged in the marketplace.

Note that the proposed remedies would still guarantee Microsoft the opportunity to participate in the specification of any given public interface, including Windows, by virtue of the fact that the interface would be public. Microsoft certainly would be allowed to build products that comply with a given public interface—as would all companies, by virtue of the public character of that interface. The only thing Microsoft would be prevented from doing is undermining the public interface by watering it down (making slightly incompatible products) and/or drawing attention away to a proprietary alternative (as it did when it encouraged Java developers to use its own non-standard version of the Java runtime environment).

In addition, this proposal would increase competition in the marketplace by allowing other companies to offer compatible products. By making Microsoft's Windows API a public standard owned by an independent entity other than Microsoft, any company would be allowed to create alternative products that offer the same functionality. Moreover, by virtue of the interface being controlled by an independent body, Microsoft would not be allowed to change the interface more rapidly than other companies could develop new products; changes to the interface would occur at a pace decided by the independent body and contributing members of the public.

How could this suggested remedy—that those of Microsoft's computer interfaces that have become *de facto* standards be made public and that Microsoft be prevented in the future from changing them at will—be enforced against Microsoft? The answer is that the District Judge Kollan-Kotelly, as part of her decision imposing remedies against Microsoft, could retain jurisdiction over the case rather than closing it completely when she issues her decision. If Microsoft should violate any of the terms of her judgment in the case, the Justice Department could call this fact to the attention of court and ask for a contempt order against Microsoft or other relief, thereby compelling Microsoft to continue to abide by the judgment of the District Court.

Could this proposal put Microsoft out of business or make the company a minor player in the computer industry? We do not think so. Microsoft has three major strengths on its side that no other firm in the world has: (1) the ability to shift markets as it pleases, (2) an astounding number of talented software developers who know how to build system-level software, and (3) the deepest pockets in the computer industry. In addition, Microsoft spends a large amount of its revenue on research, which endows the company with excellent perspective on the future of computing; though this last point gives the company an edge over many of its competitors, it by no way gives Microsoft an exclusive edge. Our proposal would only take away one of Microsoft's strengths: the ability to shift markets as it pleases. The company would still have financial resources and human resources that outweigh any competitor, and its commitment to research should keep the company abreast of any new technologies. Moreover, there is a perfectly good precedent

to turn to: Intel makes the computer chips on which Windows runs, and the company has several competitors that manufacture compatible chips. Though Intel has had to face fierce competition, it has responded with rapidly increasing processor performance and thus has continued to own a lion's share of the PC-chip market. One could argue that the performance increases of the last decade would have been far less dramatic than they have been, had Intel faced no competition. If our proposed remedy is implemented and competition is returned to the PC operating systems market, it is likely that Microsoft, like Intel, will have a "brand name" advantage in that it will create the gold standard by which all reasonably interchangeable products will be judged. After all, any Windows-compatible operating system would be judged on its ability to emulate Windows, while the reverse would not be true.

Could a small company abuse this proposed remedy? Could it build a product, declare its product's interface to be the *de facto* standard, and thereby prevent any form of competition from Microsoft? Not really: One cannot mandate that one's product be a standard; it must be demonstrated to be the most popular interface among alternatives for a given function. Even if the interface were the *de facto* standard, Microsoft would still be allowed to offer compatible products that comply with the interface. Non-monopolies would be allowed to build and market products with alternative interfaces, and this would certainly happen if the original interface was a poor design and became the standard simply by being the first to market. If any alternative interface superseded the original public one, then that alternative would become the *de facto* standard, and Microsoft would be allowed to build compatible products for it. Could Microsoft ever be put in the position wherein the *de facto* standard for some function is proprietary, the interface is protected in some way by intellectual property rights, and the owner of that interface chooses not to license its technology to Microsoft? This is conceivable—but the authors are aware of only two companies that frequently and selectively deny access to computer interfaces that are proprietary and *de facto* standards; both of those companies are monopolies; and one of the companies is Microsoft itself (the other is Intel).

Our proposed remedy prevents a monopoly from subverting public interfaces. It does not prevent a monopolistic company from competing in the marketplace; in fact, it encourages competition on the basis of good product design. The proposed solution would simply prevent a *type* of competition that currently allows large corporations to make small ones irrelevant instead of having to compete head-to-head with them.

Non-Solutions to the Problem

Given this list, then, it is easy to evaluate potential solutions to the problem, producing a number of potential fixes that would not actually solve anything:

- **Reorganize the company into an operating systems company (responsible for Windows) and an applications company (responsible for all other Microsoft software, e.g. Office, Visual Basic, Passport, etc.)**. It is likely that this will not happen anyway, due to the current direction of the case. This remedy was ordered by Judge Jackson of the District Court probably because it would tend to slow down the rate at which the Windows API changes—this would be the case because Microsoft's applications company would

not be able to exploit those changes any more quickly than other companies. Therefore, the proposal would likely eliminate abuses stemming from action #3. However, the remedy would not prevent abuses stemming from Microsoft's actions #5 or #6—subverting the Java interface standard and attempting to subvert the HTML standard.

- **Shorten the length of time a technology patent is in effect**. This is not satisfactory because Microsoft's behavior #3—changing the Windows API to make existing software incompatible—would still be legal. In addition, any patent-holder, including Microsoft, could periodically change its product with the intent to acquire a new and different patent on the product's implementation, effectively extending the lifetime of a patent indefinitely. This remedy would also not prevent actions #5 or #6.
- **Force Microsoft to divulge all (hidden) interfaces**. This addresses behavior #4—Microsoft's decision to selectively disseminate its Windows API changes—and is covered by Judge Jackson's interim provisions. However, a company such as Microsoft can adhere to the letter of the law and still disobey its spirit by changing its interface more rapidly than other companies can keep up, and Judge Jackson's Judgment would not prevent this from continuing to occur. With a large programming staff, Microsoft can breed new interfaces faster than other companies—especially smaller companies with modest-sized staffs of programmers—can write software for the old ones. This would keep a targeted company in the perpetual state of being prevented from releasing software for the latest version of Windows. This remedy would also not prevent actions #5 or #6.
- **Force Microsoft to pay a simple fine (e.g. to government agencies)**. Clearly, this would merely be a slap on the wrist to the world's most powerful software company.
- **Force OEMs to include alternative browsers and/or office software on the computers they ship**. This would still allow Microsoft to produce hidden features in its operating system directed at its own browser and/or office applications, giving them an advantage over the applications of other companies. It would prevent none of the abuses we have described.
- **Prevent Microsoft from embedding its browser technology into its operating system**. Like the other non-solutions, this would not solve the problem. It would ensure that competition exists to a degree in the browser market, but this competition would only exist insofar as Microsoft is prohibited from the maneuvers #3, #5, and #6. Moreover, it would do nothing to foster competition in the operating system market.

Sidebar: Why Version 7.0 of Internet Explorer Won't Work, But Version 7.0.1 Will

The settlement agreement of November 2nd, 2001 and the revised agreement of November 6th contain a number of proposed remedies that are also obvious non-solutions to the problem. There are so many ways that Microsoft can adhere to the letter of the agreement but still perform acts of monopoly-continuation identical or nearly identical to

those of the Browser War that their descriptions would fill an entire article by themselves. Here are, very briefly, a few of those loopholes.

The strongest remedy in the settlement agreement is an attempt to force Microsoft to divulge fully and equally to all the Windows API so that (a) there remain no "undocumented" features of the operating system and (b) no party is excluded from learning the particulars of the API so as to hinder product development or product release. The wording of the requirement is as follows: "Microsoft shall disclose … for the sole purpose of interoperating with a Windows Operating System Product, … the APIs and related Documentation that are used by Microsoft Middleware to interoperate with a Windows Operating System Product." This simple remedy has been watered down by so many mechanisms that it is all but useless:

- "For the sole purpose of interoperating with a Windows Operating System Product" means that Microsoft does not have to divulge the APIs to anyone who intends to write their software for other platforms.
- The only portion of the Windows API that Microsoft must divulge is that portion used by "Microsoft Middleware." Microsoft Middleware has been defined in the document to mean software that is distributed separately from Windows. Therefore those APIs used by any software shipped with the operating system need not be divulged. Very little of Microsoft's software is shipped independently of the operating system.
- The only portion of the Windows API that Microsoft must divulge is that portion used by "Microsoft Middleware." Microsoft Middleware has been defined in the document to mean software that is "Trademarked." Trademarked has been defined to exclude all software that has the name "Microsoft" or "Windows" in it. Therefore, Microsoft must divulge APIs only for Microsoft software that does not include the name "Microsoft" or "Windows," so those APIs used by Microsoft Internet Explorer need not be divulged.
- The only portion of the Windows API that Microsoft must divulge is that portion used by "Microsoft Middleware." Microsoft Middleware has been defined in the document to mean software that provides the same function as a "Microsoft Middleware Product." Microsoft Middleware Product has been defined to include only the following software products, most of which are shipped with the operating system: Internet Explorer, Microsoft's Java Virtual Machine, Windows Media Player, Windows Messenger, and Outlook Express. The list specifically does not include Microsoft Office; therefore, that portion of the Windows API used by Microsoft Office need not be divulged, unless it is used by the other products as well. Moreover, Microsoft can craft its "Middleware" software so as to use an arbitrarily small portion of the operating system's interface, in particular avoiding that portion used by Office, so that no software vendor could attempt to write a software product that competes with Office.
- The only portion of the Windows API that Microsoft must divulge is that portion used by "Microsoft Middleware." Microsoft Middleware has been defined in the document to include code that controls

Microsoft Middleware—in other words, Middleware is defined in terms of itself. Therefore, a good lawyer could argue that Middleware is a meaningless term, and, by implication, Microsoft would therefore be required to divulge none of the Windows API whatsoever.

- Lastly (this is the personal favorite of the authors), "Microsoft Middleware" has been defined in the document to mean all the above, subject to the following: "Software code … shall not be deemed Microsoft Middleware unless identified as a new major version." This means that, even if Microsoft is thwarted from exploiting all of the other above loopholes, the company need only divulge that portion of the Windows API that is used by version 1.0, version 2.0, version 3.0, etc. of its Middleware. Therefore Microsoft can ship a crippled version of Internet Explorer 7.0 that uses *none* of the Windows API, along with similarly crippled versions of all its other Middleware products, and then ship an immediate bug fix in version 7.0.1 that uses the entire API. Microsoft would not have to divulge any of the Windows API at all.

Again, the agreement is riddled with holes such as this—these holes are all found in one paragraph alone, so one can imagine how useless and unenforceable the entire agreement is. Perhaps the most telling feature of the agreement is one of the document's last lines in which "Windows Operating System Product" is defined thusly: "The software code that comprises a Windows Operating System Product shall be determined by Microsoft in its sole discretion." In other words, if Microsoft wants to put a ham sandwich into Windows, it has free license to do so.

Legal Precedents

The federal government has the power, and has used it in the past, to regulate an industry in such a way as to require common standards of interoperability, rather than allowing individual corporations to create their own incompatible standards. Allowing the creation of incompatible standards allows one company in an industry to eliminate the availability of reasonably interchangeable services, thereby precluding direct competition. There are many examples, but, for illustration purposes, one is the setting of standard rail specifications.

Rail Track Width

The Constitution of the United States grants to the federal government the power to regulate interstate commerce. Congress therefore has authority to legislate in this area, and, among other things, Congress may regulate competition that takes place in interstate commerce. Congress has enunciated the "rail transportation policy" of the United States to include, among others, the following goals:

> (1) to allow, to the maximum extent possible, competition and the demand for services to establish reasonable rates for transportation by rail.
> …
> (5) to foster sound economic conditions and to ensure effective competition and coordination between rail carriers and other modes [of transportation].
> …
> (12) to prohibit predatory pricing and practices, … [6, section 10101]

The construction of a second or duplicate rail line may be prevented when it is not necessary and when it would depend for its revenue on taking traffic away from the preexisting line [4]. Competition in such a case would not be beneficial to the health of the national rail system. Allowing the duplicate line would create a different sort of competition as would be the case if the different carriers were forced to provide service over the same rails.

Congress may act on its own in regulating commerce, or it may delegate to a federal agency the power to regulate a particular industry. The railroad industry, for example, was placed under the jurisdiction of the Interstate Commerce Commission when that agency was created in 1887. In 1995, Congress abolished the ICC as such but established a new agency, the Surface Transportation Board, and transferred to that new agency much of the authority over the railroad industry that previously had been exercised by the ICC.

The Supreme Court of the United States held [5] that, under the Interstate Commerce Act, a railroad carrier owed to shippers of freight in its possession destined for points beyond the limit of its line, a duty to deliver that freight to a connecting line for further transportation to reach its destination. Also, the connecting line has a corresponding duty to receive and to carry that freight to the next carrier or to its ultimate destination if that destination was along that rail line. The Court indicated that, under that Act, the federal government could require the interchange of traffic between respective railroad lines and could require that those lines establish connections where such interchange may take place.

A rail carrier is required to provide facilities for the interchange of traffic and freight. The Supreme Court has declared that the ICC (and now the STB) has the authority to require the interchange of traffic, including not only trackage, but terminal facilities as well [7]. Congress also has required facilities for the interchange of traffic:

> A rail carrier providing transportation subject to the jurisdiction of the Board … shall provide reasonable, proper and equal facilities that are within its power to provide for the interchange of traffic between, and for the receiving, forwarding and delivery of passengers and property to and from, its respective line and a connecting line of another rail carrier … [6]

Title 49, section 11102(a) of the United States Code provides that the Surface Transportation Board "may require terminal facilities, including main-line tracks for a reasonable distance outside of a terminal … to be used by another rail carrier if the Board finds that use to be practicable and in the public interest without substantially impairing the ability of the rail carrier owning the facilities … to handle its own business …" Section 11102(c) provides that the Board "May require rail carriers to enter into reciprocal switching agreements … where necessary to provide competitive rail service."

Title 49, section 11103 reads as follows:

> On application of the owner of a lateral branch line of a railroad, or of a shipper tendering interstate traffic for transportation, a rail carrier subject to the jurisdiction of the Board shall construct, maintain, and operate, on reasonable conditions, a switch connector to connect that branch line or private side track with its railroad … when the connection
> (1) is reasonably practicable;

(2) can be made safely; and
(3) will furnish sufficient business to justify its construction and maintenance.

Before 1862, railroads in this country ran on tracks of different widths, or gauges. We quote from a letter received from an official of the Federal Railroad Administration:

> At the beginning of the U.S. Civil War there were 7 different gauges in use in the U.S. and Canada. Standard gauge accounted for 53% of the total, with 4' 10", 5' 0", 5' 6", and 6' 0" accounting for significant mileage. The Civil War brought to everyone's attention the problems associated with transferring people and freight from one rail system to another. The Northeast rail systems of the U.S. had used standard gauge from the beginning, because they had imported Stephenson Locomotives from England. [27]

In 1862 Congress enacted legislation to make possible the construction of the first transcontinental railroad, "for Postal, Military and Other Purposes." One of those "other purposes," obviously, was to facilitate interstate commerce. In section 12 of this statute, adopted on July 1, 1862, Congress provided as follows:

> … The track upon the entire line of railroad and branches shall be of uniform width, to be determined by the President of the United States, so that, when completed, cars can be run from the Missouri River to the Pacific Coast; … [14]

Congress was not satisfied with President Lincoln's decision with regard to the gauge of tracks for the transcontinental railroad [15], and on March 3, 1863, enacted another statue, which stated:

> Be it enacted … that the gauge of the Pacific railroad and its branches throughout their whole extent, from the Pacific Coast to the Missouri River, shall be, and hereby is, established at four feet eight and one-half inches. [16]

This Act thus established four feet eight and one-half inches as the North American Standard Gauge [27].

The Surface Transportation Board has adopted many pages of detailed regulations on the subject of tracks. Many of these are for the purpose of setting a standard or uniform width between rails for railroads in interstate commerce and therefore under the Board's jurisdiction. These requirements specify that the rails must be no less than 4 feet 8 inches apart and no more than 4 feet 10 and one quarter inches. This makes it possible for any rail car to fit any tracks. See, e.g., 49 Code of Federal Regulation ¶¶213.51 and 213.323. The stated purpose of these particular regulations is safety, but since the government has power, under the Interstate Commerce Clause, to prevent monopolistic practices, and to ensure fair competition, such regulations also could have been enacted for those reasons as well.

Let us now describe two examples to aid in making our point.

Ex. 1: Suppose, hypothetically, that there is a very well-known railroad company that owns a large fraction of the world's rail lines but that does not own the rail line connecting Boston and New York. By the fact that this company owns so many rails, it has the "brand-name" advantage of being known to nearly all railway passengers and shippers.

Suppose this company recognizes that running its cars on the New York–Boston route would increase its profits substantially, but the company does not want to pay the owner of the line for the right to use its tracks. Suppose that the large company has bought a right of way between New York and Boston and is beginning to construct tracks on the right of way. The company will depend for its revenue on taking traffic away from the preexisting line, and there is little doubt that the large company will be successful in doing so, because it is a well-known rail carrier, and, given the choice between a rail carrier that one is familiar with and a carrier that one has never heard of, any given passenger or shipper is likely to choose the familiar name.

As mentioned before, the federal government has the power to stop the construction of the duplicate line [4].

Ex. 2: Suppose, hypothetically, that there has been no rail line for many years directly connecting Boston and New York, and rail traffic must pass through other cities and travel in a roundabout way for several hundred miles more than the direct route, to carry freight or passengers from one of these two cities to the other. Many years ago there was a direct line connecting the two cities, but it has been abandoned for many years.

Suppose that the large company from the previous example has just bought the old right of way and is beginning to construct tracks on the right of way. However, instead of building rails that are the standard gauge of 4 feet, 8 and one-half inches apart, this company is placing its rails six feet apart. The company may claim that their reason for doing this is efficiency or better service, but the direct and obvious effect is to make it impossible for railroad cars of other companies, which are built for standard-width tracks, to run on the newly constructed line. Other carriers will have to unload freight and passengers and reload them on cars built for the non-standard tracks (owned by the large company), and this will make it possible for that company to earn more revenue and greater profits.

Can there be any question as to the government's authority to block the company from building wide-gauge tracks? Clearly, the government has power to require that the rails be standard width. The government has authority to prevent the construction of non-standard-gauge tracks, also to prevent monopolistic practices or anticompetitive practices, which the construction of such a line would represent. After all, railroads in interstate commerce are subject to the Sherman act, which prohibits conduct that extinguishes competition. [11, 12]

The adverse effect of monopolistic practices over several hundred miles of railroad track is infinitesimal compared to the effect that a monopolistic practice by Microsoft has on our economic system. The repercussions throughout the economy are much greater when Microsoft performs anticompetitive acts, but these hypothetical examples can be useful in explaining Microsoft's actions and their impact on competitors.

The Microsoft interface-abuse analogy is the combination of these two examples: It is as if (1) a railroad company were to build a duplicate set of tracks that (2) are also incompatible with preexisting tracks. If Microsoft were to offer compatible, alternative software, that software would compete head-to-head with other alternatives. However, by building and promoting alternative software that uses incompatible interfaces, Microsoft offers an alternative that

depends for its revenues on taking business away from other companies' products, by preventing software by those other companies from competing head-to-head with Microsoft's software. This is the case because, by virtue of the fact that Microsoft's software is incompatible with that other software, a customer must choose one or the other and cannot (easily) have both. For example, developers writing software for Microsoft's implementation of Java cannot then run that software on the Sun JVM. Developers creating websites using Microsoft's incompatible HTML control statements are forced into the situation where their website works correctly only for Microsoft's web browser.

As we have shown in the railroad example, the government is justified in forcing companies to adhere to standards of compatibility if doing so furthers fair competition. The government is similarly justified in forcing Microsoft to adhere to standards of compatibility—doing so would not only further competition, but it would also prevent Microsoft from repeating many of its recent anticompetitive actions.

Intellectual Property Considerations

Copyrights and patents constitute property in the holder of the copyright or patent, and such property normally should not be taken away by the government except for good reason and should not be taken without providing compensation to the owner for the loss of that property. There is some authority to the effect that even if the owner of intellectual property has violated antitrust laws, the courts cannot take the intellectual property away from that owner [17].

May Microsoft and other companies in the computer industry claim the protection of our copyright or patent laws for their interfaces, even when those companies have violated the antitrust laws of the country? Computer programs may be copyrighted, and innovations in the workings of computers can be patented. Some user interfaces apparently can be copyrighted if at least some literary creativity has been used in the development of that user interface [18, 19, 20].

What about the "internal" interfaces that have been the focus of this article? Are these copyrightable, patentable? Judge Jackson, in his *Final Judgment*, had no difficulty in imposing his interim remedies against Microsoft, in requiring Microsoft to disclose its internal interfaces (the Windows API) to the public [13]. Copyright and patent law posed no barrier to requiring Microsoft to give up control of its interfaces, as far as he was concerned.

Even if internal interfaces are copyrightable, they cannot be used in such a way as to violate the nation's antitrust laws. In a case involving the Morton Salt Company [21], a competitor company was using a patent as a means of restraining competition, in violation of the antitrust laws. That company sought injunctive relief against Morton Salt, to restrain the latter from allegedly violating the patent. The court ruled in favor of Morton, reasoning that a patent should not be enforced where the patentee has used its monopoly power and the patent as a means of illegally restraining competition. Microsoft has used its Windows API, its most widely used computer interface, in ways that have violated the antitrust laws, and therefore the company should not be allowed to claim the protection of our patent or copyright laws to prevent the interface from being made public property.

The author of an article in the Stanford Law Review, written in 1993, before the advent of the current Microsoft case, said in that article that although the user interface may be copyrighted, since it may involve some literary or artistic creativity, an internal interface must not be copyrighted or patented. In that article, the author foresaw the problem that has arisen in the Microsoft case. He predicted that antitrust problems would occur if such interfaces receive copyright or patent law protection:

> [N]ew hardware manufacturers may find themselves compelled to use a computer's internal interface in order to make their machines compatible with software already independently developed for the competitor's computer. Without compatibility with preexisting software, the costs and possible limited availability of software will make consumers hesitant to buy the new hardware, even if it is superior. Thus, a copyright over an internal interface may yield a monopoly over computer hardware. … As one commentator suggests, copyright protection for elements necessary to achieve compatibility would encourage large, established firms to create *incompatible* products in an effort to set new standards, which they could then monopolize. Yet the public may embrace these new standards not because they are objectively preferable to either the previous standard or alternatives possessed by smaller firms, but because of the large firms' "penetration pricing" strategy or powerful brand name recognition.
>
> [T]he law must be cautious about protecting elements necessary to achieve compatibility because even though such protection may, in some cases, serve innovation, it risks granting far-reaching, unjustified monopolies. Any intellectual property regime that covers interface elements necessary for compatibility must be able to weigh these factors in order to protect only those elements that are truly innovative enough to warrant protection. … Given the potential monopoly effects and the ambiguous innovation effects, not protecting elements necessary for compatibility, while imperfect, is preferable to overbroad, long-lived protection. Specifically, copyright should leave the following elements of software unprotected: (1) elements dictated by efficiency; (2) internal interface elements *required* to achieve compatibility; and (3) elements of user interfaces that have already become de facto interface standards. [22]

The author of the Stanford Law Review article foresaw clearly that internal interfaces must not be copyrightable or patentable, else serious abuses of the antitrust laws, such as those perpetrated by Microsoft, are likely. An internal interface that has become the standard for the industry should be considered public property. Otherwise, problems such as those caused by Microsoft in the current case before the District Court are inevitable.

Our Position

To summarize, we propose two remedial steps that would prevent future misconduct on Microsoft's part. In remedial step A we propose that those of Microsoft's most widely-used interfaces, especially the Windows API[3], should be made public and should be made universally available, without the threat of Microsoft's manipulating those interfaces to injure competition in the future. Furthermore, remedial step B proposes that Microsoft be prevented from undermining public interfaces so as to prevent future acts similar to fragmenting the Java interface.

To restore competition to the marketplace, we propose the largely monetary remedial steps C–H. These are aimed at redressing Microsoft's illegal, anticompetitive behavior towards Sun, Netscape, Intel, and others. Clearly the United States District Court has authority within current antitrust laws to impose those remedies against Microsoft.

---

3. A similar argument would suggest that the Office file formats be included in this list as well.

Whether the District Court has the power to impose steps A and B as part of the package of remedies against Microsoft under current antitrust laws is not entirely clear. The United States Court of Appeals, in its June 28, 2001 opinion, did not mention this as a possibility, and these steps were not included as a remedy by Judge Jackson in his decision. It can be argued that declaring the Windows interface public is beyond the power of the courts under present antitrust laws, and that such a remedy can only be imposed though future legislation by Congress or through administrative regulation or adjudication under future statutory authority from Congress.

On the other hand, it can be argued that the District Court does have the authority to declare the interface public as part of the array of judicial remedies against Microsoft in the case now before it. Antitrust law, after all, consists of federal statutes and the case law interpreting those statutes, particularly the body of case law generated by the Supreme Court of the United States over many years of antitrust litigation. These cases provide the District Court with a wide range of possible remedies and with a great deal of flexibility to address the problem at hand. The Supreme Court has said that in imposing remedies for antitrust violations the trial court is not limited to merely restoring the status quo. That court should compel the violator to take actions that will cure the ill effects of the violation or violations and also should take steps that will prevent the violations from continuing into the future and should deny the violators any future benefits from their wrongdoing [9, 10].

Arguably, the United States District Court has authority under present law to make the interface public as part of the remedies to be imposed against Microsoft in the present case. But, if not, Congress should enact legislation giving a federal administrative agency the authority to adopt regulations giving it the power to hold hearings regarding interfaces and the power to declare interfaces public when they become so widely used that commerce would be adversely affected if they are not declared public property.

Conclusion

Computers fail to work when their internal interfaces, the rules by which computers and computer components operate and interact with one another, are disobeyed. These interfaces are essentially sets of rules, and we have shown that simply changing these rules can have significant economic impact. It is our contention that when one is in the position of writing the rules—any rules that affect a significant number of people—then one is obligated not to manipulate those rules to further one’s own selfish interests.

The computer industry is similarly bound to the rules by which computers interoperate, and therefore any company with the ability to manipulate rules that affect a significant number of corporations has the ability to bend the behavior of those corporations to its will. Microsoft’s ownership of the Windows API represents such an ability, because a substantial fraction of the world’s hardware and software is dependent upon the specification of the Windows interface, the Windows API, and, when that API changes, all affected must update their hardware and/or software to remain compatible with Windows. Microsoft has demonstrated in the past that it is willing to use this ability to harm individual companies, and there is no indication that it will refrain from this behavior in the future. Therefore, to

prevent future anticompetitive behavior from Microsoft, the ability to change the Windows API without limit to scope or timetable must be taken from the company. This is best done by making the Windows API a public standard and handing it to an independent standards institute (such as the IEEE, ANSI, etc.) to control, and to not allow Microsoft to change the API in its own products except under guidelines established by that institute. Microsoft could be prevented from suddenly changing the interface for its products so as to stamp out competition.

The effect of Microsoft's anticompetitive behavior is that the company retains its position as industry leader not by offering the most innovative, reliable, user-friendly products available, but by ensuring that any competing software product that shows the potential to be more innovative, reliable, or user-friendly than Microsoft's products is killed in its infancy.

Ultimately, it can be argued that Microsoft is cheating every computer-user in the world out of a better computing experience by holding back innovation in the computer industry and thereby keeping consumers' expectations of its own products artificially low. Judge Jackson ended the *Findings of Fact* expressing exactly this sentiment, and the excerpt summarizes Microsoft's behavior towards its constituents very plainly:

> Most harmful of all is the message that Microsoft's actions have conveyed to every enterprise with the potential to innovate in the computer industry. Through its conduct toward Netscape, IBM, Compaq, Intel, and others, Microsoft has demonstrated that it will use its prodigious market power and immense profits to harm any firm that insists on pursuing initiatives that could intensify competition against one of Microsoft's core products. Microsoft's past success in hurting such companies and stifling innovation deters investment in technologies and businesses that exhibit the potential to threaten Microsoft. The ultimate result is that some innovations that would truly benefit consumers never occur for the sole reason that they do not coincide with Microsoft's self-interest. [2: ¶412]

We have shown that Congress has the power to enforce industry-wide standards of interoperability and that Congress has used this power in the past. We have also shown that the regulating of its corporate behavior with regards to computing standards is the only way to prevent Microsoft from performing nearly identical acts of anticompetitive behavior in the future as it did during the mid 1990's. Were Microsoft's Windows API to be named an open standard and the company to become regulated with respect to open standards, Microsoft would not be allowed to stifle and destroy other software vendors using its demonstrated ability to shift markets by manipulating the Windows API and undermining popular interfaces; the company would instead, for the first time in a long time, be forced to compete entirely on the strengths of its product.

Microsoft has engaged in violations of the antitrust laws of our country and continues to violate those laws. Judge Thomas Penfield Jackson's Findings were accurate—they showed just how relentless Microsoft's actions have been, and his decision showed that strong remedial steps are necessary to ensure compliance with the Sherman Act. The Court of Appeals agreed with his decision in it important aspects—that Microsoft was a monopoly and had violated the Sherman Act, and that preventive steps needed to be taken by the District Court, although by a different District Judge.

But Bill Gates kept denying and denying that Microsoft had done nothing wrong. He and other Microsoft officials kept denying the truth for a long enough period of time for another political party, the Republican Party, a party generally more favorable to big business than the Democratic Party, to come into power in January, 2001. President Bush appointed a new Attorney General to head the Justice Department and had the opportunity to appoint a new set of Assistant Attorneys General. So, one of the most powerful corporations in this country, a business founded and headed by a man who happens to be the wealthiest person in the world, with tremendous power and influence, and with the capacity to contribute substantially to political parties and the candidates he favors, violated the law but repeatedly denied any wrongdoing over a long enough period of time to enable another political party to come into power, with a new set of lawyers in the Justice Department, lawyers with an obviously different viewpoint when it comes to enforcing the antitrust laws.

Then the economy, in 2001, went into a serious downturn, and on September 11, 2001, our country was attacked by terrorists. In October, 2001, the newly staffed Justice Department entered into a proposed settlement with Microsoft that does not address the problems created by Microsoft's activities of the past several years and which will not prevent it from continuing to engage in the same kinds of illegal conduct in the future.

Hopefully the proposed settlement will not be signed by the states that, as of this writing, have refused to join with the federal government in attempting to settle the case. Also, hopefully Judge Kollar-Kotelly will refuse to approve and accept the proposed settlement and instead will impose meaningful and effective remedies against Microsoft. We hope that the suggestions in this paper, which would make the Windows API public property, owned and controlled by the public—will be included in any set of remedies imposed in the case.

The facts of the Microsoft case show unfortunate behavior on the part of Microsoft and its officers, and if they are not stopped from continuing to engage in that conduct, our system for regulating antitrust violations will have broken down. The facts of the case, as outlined in the *Findings* of Judge Jackson, require that significant, effective remedies be imposed. If this is not done, the system will have miscarried because Gates and Microsoft refused to admit to wrongdoing, even though it is clear that violations took place; because a different political party, with a different approach to antitrust law won the November, 2000 election and was able to replace the Attorney General with another; because the economy faltered, thereby placing pressure on the Bush administration to ease up in pursuing antitrust violations; and because the September 11, 2001 terrorist attacks required the full attention of the Justice Department and left little time for battling Microsoft. Thus, instead of reaching a decision based on the facts as developed by Judge Jackson, the District Court will have made a decision based on extraneous factors that have nothing to do with the merits of the case. Our system of justice will have failed.

References

1. United States v. Microsoft Corporation, No. 00-5212. U.S. Court of Appeals For the District of Columbia Circuit. June 28, 2001.

2. Findings of Fact, United States of America v. Microsoft Corporation, C.A. 98-1232. United States District Court for the District of Columbia, E. Barrett Prettyman United States Courthouse, 333 Constitution Ave NW, Washington DC, 20001. November 1999.

3. U.S. v. Microsoft Corp. 87 Federal Supplement, 2d Series 30. United States District Court for the District of Columbia, E. Barrett Prettyman United States Courthouse, 333 Constitution Ave NW, Washington DC, 20001. April 2000.

4. Detroit & M. R. Co. v Boyne City, G. & A. R. Co., 286 F. 540 (D.C. Mich., 1923).

5. N. Y. C. R. Co. v Talisman, 288 U.S. 239, 77 L. td 721, 53. S Ct 308(1933).

6. Title 49, United States Code Service.

7. I.C.C. v U.S., 280 U.S. 52, 74 L. Ed 163, 50 S. Ct 53 (1929).

8. Hocking Valley R. Co. v N. Y. Coal Co., 217 F. 227 (6th Cir., 1914).

9. U.S. v United Shoe Machine Corp, 391 U.S. 244, 20 L. Ed 2d 562, 88 S. Ct. 1496 (1968)

10. U.S. v U.S. Gypsum Co., 340 U.S. 76, 95 L Ed 2d 89, 71 S. Ct. 160 (1950).

11. U.S. v Delaware L. & W. R. Co., 238 U. S. 516, 59 L. Ed 1438, 40 S. Ct. 873 ();

12. U.S. v Reading Co., 253 U.S. 26, 64 L. Ed 760, 40 S. Ct. 425 ().

13. U.S. v. Microsoft Corp., 97 Federal Supplement, 2d Series 59. 67 (D. D. C. 2000)

14. 37th Congress, Sess. II, Ch. 120, 12 Stats at Large 489, 495 at section 12 (July 1, 1862).

15. Railroad Gauge: The Evolution of Railroad Standard Gauge. http://www.railway.org/railroadgauge.htm.

16. 37th Congress, Sess. III, Ch. 112, 12 Stats at Large 807 (March 3, 1863).

17. Hartford-Empire Co. v. United States, 323 U.S. 386, 89 L Ed 322, 65. S. Ct 373 (1945), clarified, 324 U.S. 570, 89 L Ed 1198, 65 S. Ct 815 (1945).

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

19. Lotus Development Corp. v. Paperback Software Intern., 740 F. Supp. 37 (D. Mass. 1990).

20. Digital Communications Associates v. Softklone Distributing, 659 F. Supp. 449 (N. D. Ga. 1987).

21. Morton Salt Co. v. G. S. Suppiger Co., 314 U.S. 488, 86 L. Ed 363, 62 S. Ct. 402 (1942).

22. Timothy S. Teter, “Merger and the Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases,” *Stanford Law Review,* vol 45, no. 1061, pp. 1061–1098, April 1993.

23. Walter S. Mossberg. “Windows XP Has Stable System; Keeps Users in Microsoft Corral,” *The Wall Street Journal (On-Line)*, September 20, 2001.

24. John Heilemann, Pride Before the Fall. Harper Collins Publishers, New York NY, 2001.

25. By Rob Pegoraro, “U.S. Settlement Leaves Microsoft More Entrenched.” *Washington Post*. Friday, November 9, 2001; Page E01.

26. Stipulation and Revised Proposed Final Judgment, United States of America v. Microsoft Corporation, C. A. 98-1232. (11/06/2001)

27. Letter to one of the authors, dated October 24, 2001, from Mark E. Yachmetz, Associate Administrator for Railroad Development of the Federal Railroad Administration, which is part of the United States Department of Transportation.

The Authors

Bruce L. Jacob is an Assistant Professor of Electrical and Computer Engineering at the University of Maryland, College Park. He received his Ars Baccalaureate, *cum laude*, in Mathematics from Harvard University in 1988, and his M.S. and Ph.D. in Computer Science and Engineering from the University of Michigan in 1995 and 1997, respectively. In addition to his academic credentials, he has extensive experience in industry—he designed real-time embedded applications and real-time embedded architectures in the area of telecommunications for two successful startup companies: Boston Technology (now part of Comverse Systems) and Priority Call Management (now part of the Sema Group). At Priority Call Management he was employee number 2, the system architect, and the chief engineer. He built the first working prototype of the company’s product, and he built and installed the first actual product as well. His systems architecture helped the company grow from start-up to a $200 million leader in its segment of the telecommunications industry. In academic research, Jacob was responsible for the cache and memory-management design of the DARPA-funded PUMA processor, which demonstrated the viability of software-managed caches for use in general-purpose systems. His work in advanced DRAM architectures is the first comparative evaluation of today’s memory technologies, and he recently received the prestigious CAREER Award from the National Science Foundation for this research. He has authored papers on computer architecture and memory systems, low-power embedded systems, distributed computing, and astronomy.

Bruce R. Jacob began his career in 1960 as an Assistant Attorney General for the State of Florida. There he represented the respondent in the United States Supreme Court in the landmark case of Gideon v. Wainwright, 372 U.S. 335 (1963). Upon leaving that office, he engaged in the private practice of law in Bartow and Lakeland, Florida, in the firm of Holland, Bevis & Smith, now Holland & Knight. Following the completion of his LL.M. degree at Northwestern University, Professor Jacob joined the faculty of Emory University School of Law, where he established the Legal Assistance for Inmates Program at the Atlanta Penitentiary. He was appointed by the Supreme

Court as counsel for petitioner in Kaufman v. United States, 394 U.S. 217 (1969). He received his S.J.D. at the Harvard Law School. While at Harvard, he served as a Research Associate in the Center for Criminal Justice, assisted in the establishment of the Harvard Prison Legal Assistance Project, and supervised the work of law students in the defense of criminal cases and in the representation of indigents in civil matters in the Community Legal Assistance Office, Cambridge, Massachusetts. Jacob subsequently served as Professor and Director of Clinical Programs at The Ohio State University College of Law, as Dean and Professor of the Mercer University School of Law and as Vice President of Stetson University and Dean of Stetson College of Law from 1981 through 1994. Presently he is Dean Emeritus and Professor of Law at Stetson. He is an author and co-author of articles on Criminal Law and Procedure, Civil Rights and Civil Liberties, and the Administrative Law of Corrections. While on sabbatical leave during 1994-95, he took courses in the LL.M. program in Taxation at the University of Florida College of Law, and received that LL.M. in 1995. He has taught courses in criminal law as well as in tax law, administrative law, and state constitutional law.