Volume 10 | Number 2 | Autumn 2014

The Antitrust Analysis of Rules and Standards for Software Platforms

By David S. Evans

www.CompetitionPolicyInternational.com

The Antitrust Analysis of Rules and Standards for Software Platforms

BY DAVID S. EVANS[1]

Software platforms anchor vast global communities of users, application developers, device manufacturers, content providers, advertisers, and others. They drive innovation by enabling entrepreneurs, often anywhere in the world, to develop "applications" and to reach all the users of the platform, often anywhere in the world. These applications are sometimes the foundation of substantial businesses. The value of these software platforms, and their ability to support large communities, depend on the ability of the platform to promote positive externalities and reduce negative externalities. Software platforms usually impose rules and standards and often exclude participants that harm others in the community, and reward participants that benefit others in the community. Competition policy should presume that these governance systems, and the restrictions they place on platform participants—including their possible expulsion from the platform—are efficient and pro-competitive. Software platforms could, however, employ governance systems to foreclose competition. These restrictions, therefore, should not be lawful per se. Rather, courts and competition authorities should employ screens to protect pro-competitive restrictions and isolate anticompetitive ones. The application of these screens should be neutral to the licensing model chosen by the software platform creator. There is, in particular, no basis for imposing limitations that are, in effect, tougher on software platforms that use an open-source license model than on software platforms that use a proprietary license.

I. INTRODUCTION

Many people use software platforms and the applications that run them during much of the day for work and leisure. They run our mobile phones, computers, and videogame consoles and are behind our social networks. New ones are behind innovations in payments, transportation, health and fitness, connected homes, and connected shopping— all of which are transforming how we live.

Software platforms create value by providing an environment in which many different types of economic agents can benefit.[2] These economic agents typically include end users, application developers, and hardware makers. They may also include advertisers, content providers, and other economic agents depending on the platform and the business model the platform has adopted.

SOFTWARE PLATFORMS CREATE VALUE BY PROVIDING AN ENVIRONMENT IN WHICH MANY DIFFERENT TYPES OF ECONOMIC AGENTS CAN BENEFIT.

There are positive externalities between these different groups. More demand from any one group of economic agents usually increases the value of the platform to the other groups of economic agents. As with

other multi-sided platforms there are positive indirect network effects: more applications leads to more end users, which leads to more interest from application developers and hardware makers, and so forth.

Software platforms also have conflicts between participants that create negative externalities, or limit positive ones, and thereby reduce the private and social value of the platform. Participants, for example, may make decisions that fragment the platform and thereby reduce the number of participants that can interact with each other. Software platforms often adopt standards, rules, and enforcement mechanisms to deal with externalities among platform participants. Such governance systems can increase the value of the platform to participants, and to the platform owner, by curtailing negative externalities and promoting positive ones. It is possible, however, that platforms could adopt rules that harm third parties without countervailing increases in the value of the platform through reduced externalities.

This article examines how externalities can affect the social and private value of software platforms, the role of governance systems in dealing with these externalities, principles for assessing whether rules and standards are pro-competitive attempts to deal with these externalities or efforts to harm competition, and the use of screens to minimize error costs on the part of competition authorities and courts.

II. SOFTWARE PLATFORM BUSINESS MODELS AND GOVERNANCE

The crown jewel of a software platforms business is its code. Most widely used software platforms have intellectual property rights over that code based on copyright, patent, and trade secret laws. Some software platform providers enforce their claimed property rights vigorously. Others have chosen to cede certain aspects of their intellectual property rights. Their creators have decided to license the code under an open-source license that allows software "to be freely used, modified, and shared."

Governance of positive and negative externalities has proved important for software platforms regardless of the intellectual property rights and software development approaches these platforms have followed. Available governance tools differ. Fully proprietary platforms can control externalities through contracts, enforcement of intellectual property rights, and platform design. Open-source platforms are limited by their overall governance structure. That structure can range from informal to hierarchical.[4] Open-source projects, for example, can be managed loosely by a small group of volunteer programmers, a "benevolent dictator" who many choose to follow, or a for-profit company that influences the direction through funding and other decisions.

GOVERNANCE OF POSITIVE AND NEGATIVE EXTERNALITIES HAS PROVED IMPORTANT FOR SOFTWARE PLATFORMS REGARDLESS OF THE INTELLECTUAL PROPERTY RIGHTS AND SOFTWARE DEVELOPMENT APPROACHES THESE PLATFORMS HAVE FOLLOWED.

Some software platforms have adopted a hybrid of proprietary and open-source software. The for-profit company invests in the development of the software platform but provides an open-source license to the software platform. It thereby loses some control over the intellectual property rights to its platform but derives benefits from the open-source process for debugging and improving the platform. It also loses some of its ability to govern externalities on the platform.

A. Positive Externalities for Software Platforms

The primary source of value for a software platform comes from its use as a standard by end users, software developers, hardware makers, and other economic agents who can benefit from interacting with potential counterparties. Different versions of a software platform can, however, evolve in ways that reduce the ability of economic agents that use one version of the software platform to interact with economic agents that use another version of the platform. This "fragmentation" works just the opposite to standardization. It deters, rather than promotes, positive externalities.

A soft form of fragmentation can result from the interrelated decisions by the software-platform owner to release new versions of the software and the decisions by users, hardware manufacturers, and wireless carriers not to upgrade to that version. Even if the platform owner makes the platform backward compatible, applications and hardware written for the most recent version may not work with older versions.

A hard form of fragmentation can result from decisions to create a version of the software that is not compatible with other versions. That could occur as a result of a proprietary company deciding that backward compatibility imposes too much of a constraint and releasing a version that is not compatible with some existing applications and hardware. More commonly, hard fragmentation occurs under the open-source model when a software platform "forks" into multiple incompatible development efforts with different platform leaders.

Fragmentation imposes costs. Developers may have to write multiple versions of the same application to make sure that the application works the same way for all users of the various fragmented flavors of the platform. That may involve anything from trivial to wholesale changes in the code depending on the nature and degree of fragmentation. The incremental cost for writing applications for different versions may lead developers to limit themselves to writing applications for the most popular versions of the software platform.

Given the fixed cost of writing software applications some developers may decide not to write for a software platform at all if they cannot reach enough users with a single version of their applications. The software platform secures smaller positive network effects as a result of fragmentation leading to fewer compatible applications and thereby reduces the private and social value of a platform. Fragmentation also makes the platform less competitive with more standardized platforms or closed platforms that can provide greater value to members of all sides at no higher cost.

Fragmentation is a more serious problem for software platforms that use an open-source license. Proprietary software-platform owners can manage fragmentation by: (i) ensuring backward compatibility, (ii) utilizing the copyright and patent legal regimes to prevent modifications, (iii) denying access to source code, (iv) using pricing and contracts with third parties to discourage fragmentation, and (v) fragmenting their own platform only when the benefits exceed the costs. Open-source software platform owners, however, typically allow developers to modify and distribute software sometimes on the condition they do so under the same open-source license, but often not). When the initial software-

FRAGMENTATION IS A MORE SERIOUS PROBLEM FOR SOFTWARE PLATFORMS THAT USE AN OPEN-SOURCE LICENSE

platform creators decide to release their software under an open-source license they make it possible for some parts of the developer community to decide to take the software platform in a different direction than either the creator, or other parts of the developer community, would like or could anticipate.

B. Negative Externalities and the Lemons Problems

Software platforms typically rely on the interaction of users, developers, hardware makers, content providers, advertisers, and other groups to generate positive network effects and platform value. Members of these groups can, however, also impose harm on other platform participants in their same group or in other groups.

Offensive material is a common problem for internet platforms. MySpace, for example, became a "vortex of perversion" because of the type of people it attracted and the content that was posted.[5] This discouraged advertisers concerned about the possibility of being on the same web page as offensive content and thereby reduced the private value of the network.

Software platform participants also encounter the spate of problems that afflicts commerce generally. Sellers of complementary products such as applications or hardware may, for example, misrepresent their products, engage in various scams, or make it difficult to cancel recurring payments. Buyers may engage in fraudulent behavior as well.

Economic agents that provide complementary goods can also create a "lemons problem" for software platforms. The classic story involves the collapse of the Atari game console business in the early 1980s. Atari used a game cartridge that was an open standard making it possible for third parties to write games. Consumers could not observe the quality of a game until they played it. The availability of reviews was much more limited than it is today. A flood of low-quality games appeared and contributed to the rapid decline of this pioneering game company. The successful game console companies such as Sony (for its PlayStation) that followed Atari limit the ability of third parties to publish games for their platforms and impose quality controls.

ECONOMIC AGENTS THAT PROVIDE COMPLEMENTARY GOODS CAN ALSO CREATE A "LEMONS PROBLEM" FOR SOFTWARE PLATFORMS.

C. Competition Among and Between Software Platform Ecosystem

As with all products software platforms can differentiate themselves by price and along a variety of dimensions to appeal to various groups of heterogeneous consumers. But, as with all multi-sided platforms, they can also differentiate themselves by the pricing structure, which determines the relative participation of the various sides, as well as through a variety of business and design decisions that can result in differentiation of each of the sides.

Software platforms owners, in particular, can: choose whether to integrate into a combined hardware and software platform or to make themselves open to hardware makers; decide on the software platform

features to provide hardware makers, application developers, and other users of the APIs; determine the extent of possible differentiation or standardization across hardware makers and application developers; and devise rules and regulations for platform participants. Software platform owners can also decide whether to differentiate the platform itself by providing multiple versions of the platform with different features.

These decisions concerning differentiation result in externalities because, by influencing demand by members of one group on the platform, they affect the demand by members of the other groups on the platform. Differentiation could result in positive externalities by increasing demand by one group and thereby benefitting other groups. For example, differentiation of hardware could result in more users, which could thereby benefit providers of applications, which in turn would benefit users and hardware makers.

Differentiation could also result in negative externalities by reducing interoperability between members of the same or different groups. For example, differentiation of hardware could make applications incompatible across types of hardware thereby raising the costs for application developers who would react by reducing the supply of applications and raising the costs to users who would then become less likely to use the platform.

Software platform owners must account for these tensions between externalities and differentiation to maximize the value of their platforms. Owners that have made their software platform available through an open-source license, however, encounter more difficulties in managing the tradeoffs between externalities and differentiation than do owners that have secured and enforced traditional intellectual property rights. Under standard open-source licenses, hardware makers and even application developers are not intrinsically constrained by copyright or patent regimes not to make modifications to the software platform code as they are with proprietary systems. These modifications could make some hardware and software incompatible. Moreover, developers could provide alternative and potentially incompatible versions of the software platform.

SOFTWARE PLATFORM OWNERS MUST ACCOUNT FOR THESE TENSIONS BETWEEN EXTERNALITIES AND DIFFERENTIATION TO MAXIMIZE THE VALUE OF THEIR PLATFORMS.

For software platforms, fragmentation raises particularly serious concerns over negative externalities that could reduce the value of the platform overall.

III. RULES AND STANDARDS FOR REGULATING EXTERNALITIES

The value of software platforms to their owners, and to their participants, depends on the extent to which the software platform can generate positive externalities and limit negative ones. The relationship between value and externalities creates powerful incentives for software platforms to control these externalities. Proprietary software platforms, motivated by profit, have developed governance systems to harness externalities to maximize the value of their platforms. Successful open-source software platforms have also developed governance systems to deal with positive and negative externalities.

In both cases, the governance systems typically consist of "standards" (norms for the platform established by custom or by design), "rules" (prohibitions of, or requirements, for specified behavior), and "enforcement" (punishment for violations and rewards for good behavior). To examine the prevalence and nature of governance systems for software platforms I examined 15 significant software platforms, as shown in Table 1. Almost all of these platforms have governance systems that involve standards, rules, and enforcement.[6]

The standards for participants arise from design decisions and requirements that software developers and hardware makers follow to comply with given parameters. Facebook provides highly structured methods for people to communicate with their friends and Apple has a highly structured hardware environment for users and applications.

Rules specify things that application developers or hardware makers must do to meet various compatibility requirements and things that they are proscribed from doing. A number of the platforms that involve user interactions also have a variety of community rules such as those involving obscene language, pornography, and hate speech.

Almost all the software platforms have enforcement mechanisms, including expulsion. Proprietary platforms typically identify rules and enforce those rules by contracts. The open-source and hybrid platforms, as well as some of the proprietary platforms, enforce rules through a combination of compatibility tests and trademark restrictions.

ALMOST ALL THE SOFTWARE PLATFORMS HAVE ENFORCEMENT MECHANISMS, INCLUDING EXPULSION.

Table 1: Survey of Software Platforms

	Proprietary	Open Source	Hybrid
PC and Game Console	Sony PlayStation Windows	Linus	
Internet	Facebook Salesforce Tencent	Bitcoin Firefox OpenStack	Ripple
Mobile Device	Apple iOS Windows Mobile	Tizen	Android Ubuntu

To provide a deeper understanding of the role of governance systems in regulating positive and negative externalities for software platforms the remainder of this section discusses three software platforms in detail. Each software platform raises different issues that provide insights for the competition analysis in the next section.

A. The Android Operating System

Google established the Android Open Source Project ("AOSP") to coordinate the development of the software platform. In practice, Google is almost solely responsible for planning each new version of the software platform

and writing the code. It then releases that to the open-source community, which can debug and improve it as desired.

By providing a free high quality operating system to device makers and a convenient Java-based framework to developers Google secured rapid ignition and growth for Android. Of the 296.6 million smart phones sold in 2010, 67.2 million had the Android operating system. By mid 2014, there were more than 84 hardware companies that made Android handsets, thousands of developers who had published 1.3 million applications,[7] and more than 1 billion users of Android-based phones and applications.[8]

The open-source model helped drive adoption but also resulted in fragmentation that limited the value of the platform. An August 2014 study by OpenSignal found, for example, that:

> Android devices come in all shapes and sizes, with vastly different performance levels and screen sizes. Furthermore, there are many different versions of Android that are concurrently active at any one time, adding another level of fragmentation. What this means is that developing apps that work across the whole range of Android devices can be extremely challenging and time-consuming.[9]

THE OPEN-SOURCE MODEL HELPED DRIVE ADOPTION BUT ALSO RESULTED IN FRAGMENTATION THAT LIMITED THE VALUE OF THE PLATFORM.

There were 18,796 distinct Android devices in 2014. This fragmentation results in significant costs and barriers to entry by developers. Each separate device presents a risk that an application will not work properly. Another form of fragmentation results from different devices running different versions of the Android operating system. Six versions of the Android OS accounted for at least a 10 percent share of all devices with the most popular version accounting for 26.5 percent. By contrast, 91 percent of Apple mobile devices had the most recent version of the iOS as of August 2014, which was iOS 7; eight percent had the next oldest version and only one percent had earlier versions.

Over time Google has developed standards, rules, and enforcement mechanisms to deal with fragmentation and other externalities. It has developed a set of compatibility standards for hardware makers, which it launched in 2007. It has also provided various tests (also on a free and open-source basis) that hardware makers can use to ensure that their devices are compatible. Google only permits Google Play (Google's mobile applications store) and certain Google apps to be preloaded onto devices that pass these compatibility standards, which creates an additional incentive for OEMs to offer compatible devices. Google also has developed set of compatibility tests for the application developers.

In addition, Google has used innovation to reduce operating system fragmentation. It delivers important updates to users through Google Play Services, which offers a set of common APIs regardless of the version of Android a device is running.

Google ensures consistent "out of the box" functionality that consumers expect by licensing its Google Mobile Applications suite ("GMS"), which includes Google Maps, YouTube, and a few other apps. As part of

this license Google requires that the applications appear in particular screens and places on the mobile device. Google has argued that providing the GMS suite helps Android device makers compete with Apple and Windows phones, which also come preinstalled with software that consumers expect.

Google also operates a store for downloading Android applications. It imposes various quality controls on developers who want to place their applications in the store. It can remove those applications that violate the terms of Google Play's Developer Program Policy. That policy prohibits applications from a variety of activities including making modifications to the user's device, reordering default presentations of apps or settings, or engaging in various kinds of malicious and deceptive behavior.

B. The Bitcoin Digital Currency Platform

Bitcoin is a "software-based online payment system" that was created in 2008. It is based on a software platform that uses a distributed network of servers to: (i) process bitcoin transactions, (ii) create more bitcoins, and (iii) provide a compensation mechanism for the "miners" who run the servers that process transactions and create bitcoins. The software platform was established as an open-source project. Despite the media attention Bitcoin has received, while the volume of bitcoin transactions has increased Bitcoin shows no signs of the explosive growth that successful multi-sided platforms had early on.[10]

NEGATIVE EXTERNALITIES HAVE PLAGUED BITCOIN.

Negative externalities have plagued Bitcoin. The digital currency was originally conceived to handle micro-transactions digitally. This "killer app," however, provided a currency for the "dark web," where transactions are made for hard drugs, firearms, and other unsavory items. It was the currency for "The Silk Road" which was an eBay of sorts for drugs and other illegal products and services. Traditional drug cartels, and other criminal gangs, also used bitcoins for money laundering. Furthermore, a number of the exchanges and vaults lost bitcoins through either malfeasance by the operators or by cyber-thieves.

Bitcoin does not have a robust governance system for dealing with these externalities. To begin with, the Bitcoin Foundation has suffered reputational problems. Of its five original board members one was convicted of money laundering and one was the founder of the bankrupt Mt. Gox, which allegedly lost about $500 million bitcoins from hacking.

At the heart of these problems is the lack of exclusionary power—it does not have the ability to prevent anyone from using the Bitcoin platform. It can exhort; it cannot exclude. Even its power to exhort is limited. Some open-source projects such as Linus have a leader—often the original creator—whose moral authority can discipline the community. That is not the case with Bitcoin whose creator remains anonymous.

C. The Windows PC Operating System

Windows was introduced in the mid 1980s and gained widespread adoption in the 1990s. As of September 2014, more than 90 percent of computers worldwide had Windows installed. Like other software platforms

Microsoft took actions to promote positive externalities and reduce negative ones. To minimize fragmentation it made sure that each new version of Windows was backwards compatible with previous versions. As a result existing applications could work with the new versions of Windows. It also made sure that all copies of Windows it distributed provided application developers with access to the same set of features. It had software development kits ("SDKs") that instructed developers on how to develop compatible applications.

However, it did not have any general mechanisms for limiting the availability of applications based on quality or other considerations related to negative externalities. It also did not provide major operating system updates free of charge. That resulted in frequent use of older operating systems.

Microsoft's explicit governance efforts were directed mainly at computer manufacturers (also known as "Original Equipment Manufacturers" or OEMs.) Although the licenses are confidential there is some information available on them as a result of the 1998 antitrust case brought by the Department of Justice and various U.S. states. Microsoft prohibited computer manufacturer licensees from "removing any desktop icons, folders, or 'Start' menu entries; altering the initial boot sequence; and otherwise altering the appearance of the Windows desktop."[11] These prohibitions limited the computer makers' "flexibility and choices in configuring the PC desktop." On the other hand these restrictions ensured that end users would have a consistent experience.

Like some other platforms Microsoft also gave computer manufacturers rewards, in the form of marketing dollars, for doing certain things that generated positive externalities. These were part of the Market Development Program. They included requirements for features such as boot-times, memory allocation, and product configuration.

Overall, these prohibitions and rewards contributed to Windows becoming a standard platform for computer manufacturers, manufacturers of peripherals, application developers, internet content providers, and corporate and personal users. As is well known, some aspects were found to exclude competition in violation of the antitrust laws.

IV. COMPETITION POLICY ISSUES

As has been described above, software platforms can create large and expanding communities by harnessing positive and negative externalities among the various groups that benefit from the platform. Governance systems play a key role in promoting positive externalities and restricting negatives ones. That is seen most clearly in rules that require platform participants to follow certain design principles that ensure compatibility and interoperability among platform components. The force of a governance system ultimately depends on the ability to exclude economic agents that refuse to follow the rules from participating on the platform.

The use of these governance systems, and the exclusion of participants that violate these rules, is presumptively pro-competitive. There is a clear nexus between the rules, standards, and enforcement

mechanisms that software platforms typically use and an effort to maximize the economic value to the community through the promotion of positive externalities and the restriction of negative ones. These rules, standards, and enforcement mechanisms are used across software platforms of all sizes and are typically unconnected with efforts to engage in anticompetitive behavior. Many governance systems regulate community standards that could not remotely affect competition, such as prohibiting participants from engaging in fraudulent or malicious behavior.

THE USE OF THESE GOVERNANCE SYSTEMS, AND THE EXCLUSION OF PARTICIPANTS THAT VIOLATE THESE RULES, IS PRESUMPTIVELY PRO-COMPETITIVE.

Competition policy should therefore exercise caution in condemning the application of governance rules for software platforms. The cost of false positives is high. The software platform community would lose significant economic value if competition policy limited the ability of platform governance systems to harness externalities. Positive network effects lead to a multiplier effect for externalities—they magnify the losses from reducing positive externalities or increasing negative ones.[12]

A presumption is not a free pass and caution does not mean a blind eye. Software platforms could enlist governance rules, just as they can use other tools at their disposal, to engage in anticompetitive behavior. A core issue, for example, in the classic Microsoft case was whether the company used rules for hardware makers to foreclose a potential platform competitor. The U.S. courts decided they had. Competition authorities and courts therefore face the usual conundrum: how to balance false positives and false negatives in the face of uncertainty and incomplete information.

A. Antitrust Concerns For Platform Rules

Software platform governance rules involve unilateral non-price practices. They often concern contracts between the software platform and members of the customer groups of the platform. Competition policy would ordinarily analyze these contracts as vertical restraints. Two concerns could arise if standard market-power related thresholds are met.

A horizontal concern is that the software platform is using the governance system to exclude one or more competitors—that is, another software platform—from the relevant antitrust market. That is, the effect of the vertical restraint is on horizontal competition. A key issue for multi-sided platforms in this situation is whether a company is engaging in practices that would prevent its platform rival from securing a critical mass of platform participants and thereby obtaining positive network effects.[13] Software platform rules that deter participants from using rival platforms raise competition policy concerns for this reason.

The classic Microsoft case illustrates the horizontal issue. The U.S. Department of Justice ("DOJ") alleged that Microsoft engaged in a variety of practices to limit the emergence of software platforms, such as Netscape's

THE CLASSIC MICROSOFT CASE ILLUSTRATES THE HORIZONTAL ISSUE.

browser, that would reduce Window's monopoly power in operating systems. The D.C. Circuit Court of Appeals agreed that some, but not all, of the practices the DOJ complained about involved exercises in market power to protect the Windows monopoly that lacked offsetting efficiency rationales.[14]

A few of the practices the D.C. Circuit found unlawful were ones that this article would characterize as governance rules. The Court focused on license provisions "prohibiting OEMs from: removing any desktop icons, folders, or 'Start' menu entries; (2) altering the initial boot sequence; and (3) otherwise altering the appearance of the Windows desktop."[15] It found that these rules were anticompetitive with one exception. The third rule prohibited OEMs from automatically launching alternative interfaces. The court found that the pro-competitive benefits of that offset any anticompetitive harm.

Some software platforms also make applications, hardware devices, and other products that compete with products provided by businesses on various sides of the platform. A vertical concern is that software platforms are using governance rules to foreclose competing products and leverage their dominance in software platforms into the adjacent market for complementary products. That is, the effect of the vertical restraint is in an adjacent market.

The Microsoft case illustrates this concern as well. The government claimed that Microsoft was trying to foreclose Netscape's browser to establish a monopoly with Microsoft's Internet Explorer browser. Microsoft's standards, rules, and enforcement actions were key elements in that strategy. As discussed above Microsoft's contracts with hardware manufacturers imposed several rules concerning how they could modify the Windows desktop. The government claimed these were part of Microsoft's strategy to foreclose Netscape and other rivals from developing competing software platforms that would reduce Microsoft's operating system monopoly. The D.C. Circuit agreed that several of Microsoft's rules were on balance anticompetitive.

Horizontal and vertical concerns over governance rules ultimately turn on the same two issues under U.S. case law and under the decisional practice of the European Commission: whether the practice forecloses competition, and whether it generates efficiencies.

B. The Implications of Open-Source Licensing for Competition Analysis

The open-source licensing model raises some issues that are unique to software platforms. By design this model enables developers and hardware makers to modify the code for the software platform. The benefit of the open-source model is that it encourages innovation by allowing anyone to introduce changes. Participants in the ecosystem can decide for themselves whether those innovations are beneficial or not. The drawback of the open-source model is that it can lead to incompatible versions of the software platform and thereby reduce positive network effects. Governance systems for open-source software permit, but try to discourage, this fragmentation because it reduces positive network effects.

This tradeoff is similar to that between inter-brand and intra-brand competition. Governance rules that limit fragmentation increase the value of the software platform. They strengthen the ability of the software

platform to compete against rivals that have a proprietary model in which the platform owner has complete control over the degree of fragmentation. They therefore increase inter-brand competition. However, governance rules that limit fragmentation, if they are successful, also tend to narrow the degree of differentiation between variants of the software platform as well as the number of alternative viable versions of the platform. They therefore tend to reduce potential intra-brand competition, which competition policy sometimes frowns on.

However, governance rules for open-source software platforms are very different from manufacturer restraints on distributors in an important respect. The value of any software platform depends on the extent to which it provides a standard compatible platform for all participants. Manufacturer vertical restraints typically involve ancillary conditions, such as price and service, related to the sale of the product. Governance rules to control fragmentation are equivalent to rules that prohibit distributors from changing the features of the product they are selling.

HOWEVER, GOVERNANCE RULES FOR OPEN-SOURCE SOFTWARE PLATFORMS ARE VERY DIFFERENT FROM MANUFACTURER RESTRAINTS ON DISTRIBUTORS IN AN IMPORTANT RESPECT.

These considerations lead to an important point concerning the role of competition policy in promoting alternative software platform business models. Software platform creators have the option of choosing a proprietary, pure open-source, or hybrid proprietary/open-source model in developing and popularizing their platforms. Each of these models has its merits in terms of promoting the efficient development of platforms as is clear from the existence of successful platforms following each of these models. There is no economic reason to believe there are market failures in the selection of these alternative models by software platform creators. In particular there is no reason to believe that software platform creators are inefficiently choosing to pursue open-source licensing models rather than closed proprietary models. There is also no economic basis to believe that software platform creators are choosing open-source models for anticompetitive reasons. They obviously have no market power when they are making these decisions.

The application of competition policy should therefore be neutral across these alternative models. Courts and competition authorities should exercise care that they do not impose policies that could encourage software platform creators to choose one model over the other. Policies, for example, that restrain software platforms under an open-source license from limiting fragmentation would have the perverse, and inefficient, result of encouraging software platform creators to adopt closed proprietary platforms.

THE APPLICATION OF COMPETITION POLICY SHOULD THEREFORE BE NEUTRAL ACROSS THESE ALTERNATIVE MODELS.

C. Competition Policy Screens for Software Platform Rules

I have previously advocated for courts and competition authorities to follow a three-step test to evaluate complaints regarding an element of a governance system for a multi-sided platform.[16] The same test should be followed for software platforms to balance the costs of errors from false positives, which can result in the sacrifice of significant positive network effects, and false negatives, which can allow the continuation of anticompetitive exclusion.

The test assumes that the complainant has established a relevant market, that the software platform has market power, and that the practice has the potential to harm competition:

1. In the first step the defendant has the opportunity to establish that the practice results from the application of a governance system for dealing with externalities. If the platform cannot do so then the standard rule of reason analysis applies. Otherwise the decision-maker moves to the second step.

2. In the second step the complainant has the burden of demonstrating that the practice in question is not reasonably related to the use of a governance system to restrict negative externalities or promote positive ones. The complainant could, for example, show that the rule is a pretext for excluding competition. If the complainant cannot do so then the matter is concluded in favor of the defendant. If the complainant can make this demonstration, the analysis proceeds to the third step.

3. In the third step the standard rule of reason analysis applies. In this step complainant has the burden of showing that the practice harmed competition through foreclosure. If the complainant meets that burden the platform defendant then has the burden of showing that the practice provides efficiencies that outweigh any anticompetitive effects.

In all steps the analysis is neutral to the type of software platform. The decision-maker avoids results that, if applied generally, would discourage software platform creators from choosing a closed proprietary model, or open-source licensing model, or a hybrid model of those two.

THE TEST PROVIDES A USEFUL METHOD FOR COMPETITION AUTHORITIES TO EVALUATE COMPLAINTS AND USING THEIR PROSECUTORIAL DISCRETION IN DETERMINING WHETHER A COMPLAINT AGAINST A GOVERNANCE PRACTICE MERITS CLOSE ATTENTION.

The details of this test of course will differ depending on the case law of the jurisdiction and decisional practice of the competition authority. The case law of the jurisdiction may preclude applying this test. However, the test provides a useful method for competition authorities to evaluate complaints and using their prosecutorial discretion in determining whether a complaint against a governance practice merits close attention.

V. CONCLUSIONS

Software platforms drive innovation by enabling entrepreneurs, often anywhere in the world, to develop "applications" and to reach all the users of the platform, often anywhere in the world. They make innovation democratic, global, distributed, and decentralized. As the software platform model has progressed over the roughly four decades since the invention of the personal computer it has demonstrated its power to drive economic progress.

The value of these software platforms, and their ability to support large communities, depend on the ability of the platform to promote positive externalities and reduce negative externalities. Software platforms need governance systems that impose rules and standards and that have mechanisms for requiring platform participants to adhere to these rules and standards. They need to be able to exclude participants that harm others from the platform.

Most significant software platforms have established governance systems. On their face they restrict negative externalities and promote positive ones, thereby increasing the value of the platform to its participants. Competition policy should presume these governance systems, and the restrictions they place on platform participants—including their possible exclusion or expulsion from the platform—are efficient and therefore pro-competitive.

Software platforms could employ governance systems to foreclose competition and therefore these restrictions should not be per se lawful. Rather, courts and competition authorities should employ screens to protect pro-competitive restrictions and isolate anticompetitive ones. The application of these screens should be neutral to the licensing model chosen by the software platform creator. There is, in particular, no basis for imposing tougher limitations on software platforms operated under a pure or hybrid open-source model than on software platforms operated under a closed proprietary model. ▲

RATHER, COURTS AND COMPETITION AUTHORITIES SHOULD EMPLOY SCREENS TO PROTECT PRO-COMPETITIVE RESTRICTIONS AND ISOLATE ANTICOMPETITIVE ONES.

---

[1] Chairman, Global Economics Group; Executive Director, Jevons Institute for Competition Law and Economics and Visiting Professor, University College London; Lecturer, University of Chicago Law School. I would like to thank Madeleine Chen and Alexis Pirchio who worked closely with me on this article and Google for research funding. None of these individuals or entities necessarily agrees with me and I retain sole ownership of any errors. This Article is an abbreviated version of a working paper by the same name, available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2520860, which includes more detail and most of the source citations.

[2] Software platforms consist of lines of computer code that support different features offered by the platform. They typically make some of their features available to third party developers. They do this by creating Application Programming Interfaces ("APIs"). APIs enable third parties to access and make use of these features. Some software platforms include operating systems that manage hardware, like Windows, but many others do not, like Facebook.

[3] The existence of externalities and the need for a governance system is a general feature of multi-sided platforms. See David S. Evans, Governing Bad Behavior By Users of Multi-Sided Platforms, 27 BERKELEY TECH. L.J. 1112-1113 (2012). Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1950474.

[4] *See*, for example, CHARLES M. SCHWEIK & ROBERT C. ENGLISH, INTERNET SUCCESS: A STUDY OF OPEN-SOURCED SOFTWARE COMMONS (2012).

[5] *See* Felix Gillette, T*he Rise and Inglorious Fall of MySpace,* BLOOMBERG BUSINESSWEEK (Jun. 27, 2011), *available at* http://www.businessweek.com/magazine/content/11_27/b4235053917570.htm.

[6] Bitcoin is the major exception as it lacks clear rules and enforcement methods. For detailed results for these platforms see Evans, *supra* note 1.

[7] *See* AppBrain Stats (Oct 21, 2014), *available at* http://www.appbrain.com/stats/number-of-android-apps.

[8] *See* Christopher Trout, *Android Still the Dominant Mobile OS with 1 Billion Active Users,* ENGADGET (Jun 25, 2014), *available at* http://www.engadget.com/2014/06/25/google-io-2014-by-the-numbers/.

[9] *Android Fragmentation Visualized,* OPENSIGNALS (August, 2014), *available at* http://opensignal.com/assets/pdf/reports/2014_08_fragmentation_report.pdf.

[10] For an overview of Bitcoin see David S. Evans, *Economic Aspects of Bitcoins and Other Decentralized Public-Ledger Currency Platforms,* THE UNIVERSITY OF CHICAGO COASE-SANDOR INSTITUTE FOR LAW & ECONOMICS Paper No. 685, *available at* SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2424516

[11] Memorandum Opinion, New York v. Microsoft Corp., 224 F. Supp. 2d 76 (D.D.C., 2002) (No. 98–1233) p. 11.

[12] False negatives have costs too. Anticompetitive strategies by software platforms could prevent a rival platform or a rival complementary product from attracting users and thereby exclude these rivals from the market. The reduced competition, and perhaps the elimination of an alternative choice or even a new product, would impose losses on consumers. As a general matter false negatives do not result in the loss of positive externalities or have the multiplier effect mentioned above. Indeed, the rivals could fragment the market and thereby reduce positive network effects. The relative costs of false positives and negatives therefore also support a presumption that software platform governance systems, and their applications to the participants of the platform, are pro-competitive.

[13] David S, Evans, *Economics of Vertical Restraints for Multi-Sided Platforms,* 9(1) COMPETITION POL'Y INT'L, (Spring 2013), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2195778.

[14] For a review of the case *see* David S. Evans, Albert Nichols, & Richard Schmalensee, *United States v. Microsoft: Did Consumers Win?* 1(3) J. COMPETITION L. & ECON. 497–539 (2005). *Available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=757426.

[15] Memorandum Opinion, New York v. Microsoft Corp., 224 F. Supp. 2d 76 (D.D.C., 2002) (No. 98–1233) p. 11.

[16] This three-step process for platform governance systems was proposed in Evans, *supra* note 3 at 1247ff.