The Antitrust Analysis of Rules and Standards for Software Platforms

David S. Evans
David.Evans@chicagounbound.edu

Follow this and additional works at: https://chicagounbound.uchicago.edu/law_and_economics

Part of the Law Commons

Recommended Citation

This Working Paper is brought to you for free and open access by the Coase-Sandor Institute for Law and Economics at Chicago Unbound. It has been accepted for inclusion in Coase-Sandor Working Paper Series in Law and Economics by an authorized administrator of Chicago Unbound. For more information, please contact unbound@law.uchicago.edu.
The Antitrust Analysis of Rules and Standards for Software Platforms

David S. Evans

THE LAW SCHOOL
THE UNIVERSITY OF CHICAGO

November 2014
THE ANTITRUST ANALYSIS OF

RULES AND STANDARDS FOR SOFTWARE PLATFORMS

David S. Evans*

8 November 2014

Abstract

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, depends on the ability of the platform to promote positive externalities and reduce externalities. Software platforms usually impose rules and standards and often exclude, or bounce, 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 anti-competitive 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 that use an open-source license model than on software platforms that use a proprietary license.

* 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. An abridged version of this paper under the same title appears in Competition Policy International, Autumn 2014, Vol. 10, No. 2.
I. INTRODUCTION

Most people use software platforms and the applications that run on them constantly over the course 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. They are helping to break down the walls between the online and offline worlds as physical space becomes connected to the cloud. Some software platforms include operating systems that manage hardware, like Windows, but many others do not, like Facebook.

Software platforms are multi-sided platforms. They enable third-party developers, content providers, and hardware manufacturers to provide complementary products and attract users. They anchor vast global business ecosystems. The Android operating system exemplifies the scale. The platform participants consist of more than 84 hardware companies that make Android handsets, thousands of developers that have already written more than 1.3 million applications, and more than 1 billion people who use Android-based phones and applications.

Software platforms, as all multi-sided platforms, have conflicts between participants. These conflicts can create negative externalities or limit positive externalities. They 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. That is a particularly serious problem for software platforms that are operated under an open-source license because software developers and hardware makers are not intrinsically constrained by copyright or patent regimes not to make modifications to the platform as they are with proprietary systems.

Like other multi-sided platforms, software platforms often adopt governance systems to deal with externalities among platform participants. They adopt binding rules and standards that require participants to use compatible software and hardware technologies. They also monitor the quality of complementary products and exclude those that do not follow rules or meet standards from the platform. Platform governance rules can vary from restrictions on offensive content provided through the platform, to technical requirements on hardware that ensure that it will work with the software operating system, to requirements on hardware and software applications.

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. These rules,
however, may also restrict the actions of third parties. That can lead these third parties to bring complaints to courts or regulatory authorities. In particular they can claim that the restraints foreclose them from a market and are anticompetitive under various competition laws around the world.

The courts and regulatory authorities could simply dismiss these claims if they could presume that the rules were procompetitive efforts to harness externalities for the platform. 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. A platform could adopt a rule as a sham to bar a potential rival from a relevant market or it might adopt a rule that restricts competition more than is necessary to achieve the objectives of the rule. That was a core issue in *US v. Microsoft*, for example, in which Microsoft was accused of imposing various contract conditions, such as prohibiting computer manufacturer licensees of Windows from deleting Windows desktop icons, to exclude Netscape from the market.\(^6\) Therefore, as usual for competition policy matters, it is necessary to screen procompetitive from anticompetitive restraints in a way that minimizes error costs resulting from false negatives and false positives.\(^7\)

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 procompetitive 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. The definition of software platforms used for this article is broad. It consists of any software-based product that enables different groups of economic agents to interact. It covers a range from mobile operating systems such as Apple iOS to social networks such as LinkedIn.

Part II describes the role that software platforms play in the economy. Part III examines the role of externalities and governance systems in software platform-based ecosystems. Part IV analyzes the resulting competition issues and presents a framework for competition authorities and courts to use in screening procompetitive from anticompetitive uses of rules and standards. Part V concludes.

**II. ROLE OF SOFTWARE PLATFORMS IN THE ECONOMY**

Beginning with the introduction of the personal computer in the late 1970s, software platforms have played an increasingly important role in the economy.\(^8\) They have moved from personal welfare of the participants as well as the platform owner’s profits.

---

\(^6\) United States v. Microsoft Corporation, 253 F.3d 34 (D.C. Cir. 2001) (en banc).


computers, to the Internet, to mobile phones, and to the Internet of Things.

A. **Basics of Software Platforms**

Software platforms consist of lines of computer code. That code performs various tasks that support different features offered by the platform. Typically the platform has a user interface which enables the user to interact with the platform. It often has various features that users work with directly. Facebook users can keep track of their friends in various ways through features in that platform. Some software platforms are designed to work with particular hardware and include features that enable users to manipulate the hardware. Microsoft Windows enables users to organize files on their computer hard drives.

Software platforms 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 invest significantly in supporting third-party developers. The more applications they can encourage the more valuable their platform becomes to users and, when applicable, hardware makers. Software platform providers therefore develop features that many developers can use to create applications. They may also provide software developer kits (SDKs) and other resources to developers for this purpose.\(^9\)

Hardware makers and software platforms often work together. The hardware maker needs to make sure that platforms can support functionality included in the hardware it makes. Hardware makers sell more devices if users have more complementary products—particularly applications—that work with those products. As a result hardware makers need to make sure that there is or will be a robust supply of complements from platform participants. The software platform needs to make sure that platform users and developers can avail themselves of the features of the hardware. The software platform has code that works with various portions of the hardware such as the chip in a computing device, a particular kind of printer, and the camera for a mobile phone.

Software platforms create value by providing an environment in which many different types of economic agents can benefit and are therefore multi-sided platforms.\(^10\) 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 it has adopted. There are positive externalities between these different groups.\(^11\) More demand from any group of economic agents often increases the value of the platform to the other groups of economic agents. That thereby increases the demand by these other groups leading to positive feedback effects. End users benefit when more application developers write more

---

\(^9\) For an example see the resources for developers of applications for Facebook at [FACEBOOK DEVELOPERS](https://developers.facebook.com/docs).


\(^11\) The possible exception is advertising which may impose negative externalities in some cases.
applications. More applications leads to more end users, which leads to more interest from application developers and hardware makers, and so forth. At the same time, the success of the platform depends on its ability to limit often divergent interests of hardware and application developers. While hardware makers are interested in differentiating their products in order to capture more consumer sales, application developers want to write their code once with certainty that their apps will perform properly on any hardware. If these divergent interests are not harmonized, then hardware differentiation can lead to incompatibilities, driving applications developers away from the platform, which in turn makes the platform less attractive to users and ultimately harms hardware sales.

B. GROWTH AND INFLUENCE OF SOFTWARE PLATFORMS

Software platforms date back to the introduction of mainframe computers and programming languages in the 1950s. Mainframes had “assembly language” which enabled programmers to write alphanumeric code that was then translated into binary machine code. It was a platform for higher-level languages, such as Fortran, and programmers who used those languages. Those languages themselves became platforms, which provided a relatively standard way to write programs across different computer-hardware platforms.

The software-platform model that is the focus of this article began in the late 1970s with the Apple II. Software Arts developed VisiCalc, the first spreadsheet program, for the new Apple computer. VisiCalc was the first “killer app”—an application so popular that it dramatically increases the adoption rate of the platform. By the mid 1980s, the companies that were responsible for the software that ran personal computers—principally Apple and Microsoft during this time—recognized the importance of courting “application developers” to create products that would drive interest in using particular software and hardware platforms.

Over the next decade Apple and Microsoft increased their efforts to design software platforms in ways that reduced the cost of developing for their platforms and increased the value of the platform for developers by providing more components that they could potentially use off the shelf through APIs. Apple held its first World Wide Developer Conference (WWDC) in 1990 and Microsoft introduced Windows 3.0 that year with a rich set of APIs. By 1998 Microsoft had more

12 The positive feedback loop between applications and users was a key issue in the U.S. Department of Justice case against Microsoft over Internet Explorer (where the focus was on the “applications barrier to entry”) and the European Commission’s case against Microsoft over Windows Media Player (where the focus was on the feedback effect between content and users for media players). United States v. Microsoft, Civil Action No. 96-1232 and EUROPEAN COMPETITION COMMISSION, CASE COMP/C-3/37.792—MICROSOFT (2004).


14 DAVID S. EVANS, ANDREI HAGIU AND RICHARD SCHMALENSEE, INVISIBLE ENGINES: HOW SOFTWARE PLATFORMS DRIVE INNOVATION AND TRANSFORM INDUSTRIES Chapter 4 (MIT Press, 2006)
than 70,000 applications written for it.\textsuperscript{15} Unlike Apple, Microsoft also made its software platform available to “Original Equipment Manufacturers” (OEMs) that produced personal computers to makers. As of 1998 there were hundreds of computer manufacturers that relied on Windows.

The Internet spawned a new set of software platforms that provided products and services to end users such as social networking, search, commerce, and customer relationship management. Platform designers have provided features made available through APIs to enable developers to write applications for these platforms and to thereby generate positive feedback effects. More than nine million applications and websites were integrated with Facebook through APIs as of March 2012.\textsuperscript{16} Internet-based platforms have used APIs in a simpler way to create positive feedback effects. YouTube, for example, provided an API that made it easy for websites to embed YouTube videos.

Software platforms such as iOS, Android, and Windows Mobile are behind the rapid development of the smart mobile-wireless ecosystem. These platforms have aggressively courted developers and have developed easy methods—based around “App Stores”—for developers to provide and users to obtain applications.\textsuperscript{17} Apple had more than 1.2 million applications as of June 2014.\textsuperscript{18} Google and Microsoft have also made their platforms open to handset manufacturers. Smart mobile wireless devices are usually connected to the Internet. Some of the applications they connect to are Internet-based software platforms.

The spread of Internet-connected mobile wireless devices has resulted in the use of software platforms and applications that merge the online and physical worlds. People and businesses can use software platforms and applications in physical locations. This has resulted in a range of uses, many in initial stages, involving payments, wearables, the connected home, and more generally the Internet of Things. To take one example, the development of beacons (small Internet-connected devices that can be placed in physical locations) and mobile wireless devices can enable businesses and people to interact within and across physical stores.\textsuperscript{19} Software platforms can then work with those devices to support applications that provide value to merchants and shoppers.

The importance of software platforms for the global economy is evidenced from the numbers. Table 1 shows the number of users (or hardware units as a proxy) and applications for many of the

\begin{footnotesize}
\begin{enumerate}
\item Court’s Findings of Fact, United States v. Microsoft Corporation, Civil Action No. 96-1232, ¶ 40.
\item In material submitted for its IPO Facebook reported, “The Facebook Platform is a set of tools and APIs that developers can use to build social apps on Facebook or to integrate their websites with Facebook.” Facebook, Inc. Form S-1 Registration Statement, Securities and Exchange Commission, available at http://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm. Facebook supports applications like Farmville. It also enables websites for allow people to sign into their websites using their Facebook credentials. There were 500,000 Facebook applications as of 2010. See http://www.digitalbuzzblog.com/facebook-statistics-facts-figures-for-2010, visited October 25, 2014.
\item Tyson McCann, \textit{The Art of the App Store: The Business of Apple Development}, (John Wiley & Sons, Inc. 2011)
\item See, for example, Sarah Perez, \textit{iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads to Date}, TECHCRUNCH, Jun 2, 2014, http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/
\end{enumerate}
\end{footnotesize}
leading platforms. As of 2014, 18 percent of people globally are active Facebook users, 20 25 percent use a smart mobile phone, and most knowledge workers use Windows. 21

Table 1: Number of Users and Applications for Representative Applications

<table>
<thead>
<tr>
<th>Software Platform</th>
<th>Users</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>1.35 billion devices 22</td>
<td>4 million (as of 2010) 23</td>
</tr>
<tr>
<td>Facebook</td>
<td>1.3 billion active users</td>
<td>9 million (as of 2012) including applications and websites 24</td>
</tr>
<tr>
<td>iOS</td>
<td>800 million devices 25</td>
<td>1.2 million 26</td>
</tr>
</tbody>
</table>

C. Software Platform Business Models

Software platforms are all based on software code. This code is one of the crown jewels of the platform business. 27 Most widely used software platforms have intellectual property rights over that code from copyright, patent and trade secret laws. Examples include Windows and iOS. For-profit companies own these software platforms and enforce their claimed property rights vigorously.

Some software platforms have chosen to cede certain aspects of their intellectual property rights. Their creators have decided to license the code under an open-source license. An open-source

---


24 See note 20.


26 See Sarah Perez, iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads to Date, TECHCRUNCH, Jun 2, 2014, http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/.

27 For open-source it is not the crown jewel in the sense of deriving direct financial benefit but it is the crown jewel in the sense that is it the ultimate source of social value for the platform ecosystem.
license allows software “to be freely used, modified, and shared.” Adopters must use the same open-source license if they distribute a modified version of the software to third parties who then have the same rights to freely use, modify, and share that version of the software. Examples include Bitcoin, a software platform for money transfer, and Linux, a general-purpose computing software platform. Non-profit foundations often own the intellectual property rights that are made available under the open-source license and guide the development of the software-platform ecosystem.

Table 2: Examples of Software Platforms and Use of Open-Source License

<table>
<thead>
<tr>
<th>Proprietary License</th>
<th>PCs and Game Consoles</th>
<th>Internet-Based</th>
<th>Smart Mobile-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows, Apple OS, Sony PlayStation</td>
<td>Facebook, Salesforce.com</td>
<td>Apple iOS, Windows</td>
<td></td>
</tr>
</tbody>
</table>

An oft-cited benefit of the open-source model is that programmers from around the world develop and debug the software. These programmers often work through a decentralized and distributed network with loose coordination from the leader of the open-source project. This approach can result in innovative and robust software. The open-source saying that “given enough eyeballs, all bugs are shallow” reflects the view that with many programmers independently looking at software they can find and fix all problems. As a result, some for-profit companies have relied on open-source development by releasing software under one of the open-source licenses and relying on a mix of paid and volunteer programmers to work on the software. These software-platform owners, in effect, give up some of their intellectual property rights in return for relying on this development model. Ripple, a software-based money-transfer platform, for example, released its proprietary

---

28 See http://opendefinition.org/.
31 Lack of management of open-source software can also lead to serious defects. The widely used open source internet security software, OpenSSL, had a serious security flaw—known as Heartbleed—which was caused by one of the volunteer programmers working on the software. See, for example, Digital Heart Attack, THE ECONOMIST, April 12, 2014, http://www.economist.com/news/business/21600691-flaw-popular-internet-security-software-could-have-serious-consequences-all-sorts. Linux and Unix web servers also were found to have a serious security flaw as a result of software code that was written by a volunteer in 1992. See Steven Vaughan-Nichols, Unix/Linux Bash: Critical security hole unmasked, ZDNET, September 24, 2014, http://www.zdnet.com/unixlinux-bash-critical-security-hole-uncovered-7000034021/.
software under an open-source license. Table 2 provides examples of software platforms that have relied on an open-source license that cedes some intellectual property rights or on a traditional proprietary license that protects their intellectual property rights.

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. The tools available 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. Open-source projects, for example, are sometimes 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. Linus Torvalds, for example, is the project coordinator for Linux, attached to the Linux Foundation, and has final say on what gets incorporated into the main distribution of Linux. Developers can still decide to distribute alternative versions of Linux but not with his blessing.

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. Google did this with Android for mobile phones and Ripple did this for digital currencies.

III. THE ROLE OF EXTERNALITIES IN CREATING VALUABLE SOFTWARE-PLATFORM BASED ECOSYSTEM

There is an externality whenever the well being of one economic agent is directly affected by the actions of another economic agent through something other than prices. The externality could be positive as when a homeowner's flowers delight its neighbors or negative as when driving pollutes the air that others breathe. A private externality arises when one economic agent benefits or harms another economic agent, such as a person trampling another person's private garden. A social externality arises when an economic agent benefits or harms a broader group of economic agents as when a colleague smokes in the office.

The literature on multi-sided platforms has mainly focused on two particular kinds of positive

---

33 See, for example, CHARLES M. SCHWEIK AND ROBERT C. ENGLISH, *INTERNET SUCCESS: A STUDY OF OPEN-SOURCED SOFTWARE COMMONS* (MIT Press, 2012)
34 Id., Chapter 3.
35 See, for example, ANDREU MAS-COLELL, MICHAEL D. WHINSTON, AND JERRY R. GREEN, *MICROECONOMIC THEORY* 352 (Oxford University Press, 1995).
externalities.\textsuperscript{36} Membership externalities arise when the value of the platform to one group of economic agents increases the more another group of economic agents also use the platform. Usage externalities arise when the ability of two economic agents to obtain value depends on their both using the same platform.

This Article focuses on externalities for software platforms that arise for any reason and not just through usage and membership externalities. The value of the software platform increases through promoting positive externalities and discouraging negative externalities regardless of the source of these externalities.

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. The more agents that participate for each interdependent group the greater the value to agents in the other groups. These indirect network externalities can result in the well-known positive feedback loop. More members in group A results in greater value for members of group B which results in more members of group B joining the platform which in turn increases the value to group A and so on.

By encouraging more participation from each group, the platform owner can increase the private and social value of the platform. Multi-sided platforms often use the price structure to do this. They depress price to one group of economic agents that is particularly valuable to another group of economic agents. Software platforms, for example, often provide free, or subsidized support, to software developers to encourage them to write applications for the platform. Software platforms that support hardware makers such as Windows engage in extensive discussions with hardware makers to increase the value of new versions of the platform as well as to insure compatibility.

Software platforms for mobile devices have encouraged positive externalities between users and application developers by creating “application stores”, such as Google Play, which reduce the transactions costs between these two groups. These stores provide a convenient location where users can search for, obtain information on, and buy applications and where application developers can sell their applications.

Encouraging users to find bugs is another common way that software platforms generate positive externalities. Open source platforms encourage developers to identify bugs and provide a process for them to communicate those bugs, and in some cases provide fixes, to core developers. Proprietary platforms often release “Beta” versions of their software and encourage users to identify bugs. These efforts improve the value of the software platform for all participants.

\textsuperscript{36} The main exceptions are the articles mentioned in footnote 4.
B. **FRAGMENTATION AND EXTERNALITIES**

Different versions of a software platform can 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.

The classic case involves Unix. AT&T developed and owned the Unix software platform. It was prohibited by an antitrust consent decree with the US Department of Justice from operating in the computer industry. It therefore made licenses to Unix available for free. Over time, as developers modified the software code, multiple incompatible versions of Unix evolved. Currently, there are 10 registered product versions of Unix 03 and 4 registered product versions of Unix 98. These multiple versions limited Unix from securing positive network effects as applications and hardware that worked with one version of Unix did not work with others.

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. Microsoft, for example, has a compatibility center that enables Windows users to determine which hardware and software products are compatible. A related problem occurs when hardware manufacturers do not provide certain features, such as sufficient memory, that application developers need to make use of the platform.

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. Microsoft made that choice when it migrated from DOS to Windows. More commonly, hard fragmentation occurs under the open source model when a software platform “forks” into multiple incompatible development efforts with different platform leaders. Amazon, for example, uses a forked version of Android for its Amazon Fire phone. Some Android apps that work on most other Android phones may not work with Amazon Fire phones. Amazon Fire phones are not tested and certified to meet the

---

37 See, for example, UNIX Past, THE OPEN GROUP, http://www.unix.org/what_is_unix/history_timeline.html
38 The Open Brand, register of Certified Products, The Open Group, http://www.opengroup.org/openbrand/register/
40 See http://www.washingtonpost.com/business/technology/android-phones-vulnerable-to-hackers/2013/02/06/f3248922-6723-11e2-9e1b-07db1d2cc9b_story.html
42 The full Google suite of apps do not work on Fire. This includes Gmail, Drive, Google Maps, Google Calendar, YouTube, and Google+. See http://www.cnet.com/news/apps-on-the-amazon-fire-phone/. Other Android popular apps that do not work on Fire are: Snapchat, Beats Music, Viber, NYTimes, Starbucks, Weather Bug. See
compatibility standards set by the Open Handset Alliance for Android.  

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 platform. That may involve anything from trivial to wholesale changes in the code depending on the nature and degree of fragmentation. Because of the incremental cost for writing applications for different versions developers may decide to limit themselves to writing applications for the most popular versions of the software platform. Given the fixed cost of writing software applications developers may decide not to write at all for a software platform if they cannot reach enough users with a single version of their applications. Fragmentation can therefore result in an entry barrier for at least some application developers and a loss of potential positive externalities. Developers obtain access to a smaller group of users and users obtain access to a smaller group of applications because of these interrelated decisions by members of the software platform ecosystem. The software platform secures less positive network effects than it would in the absence of fragmentation. All else equal, fragmentation reduces the private and social value of a platform, which has lower positive feedback effects because the platform has fewer members on all sides. Fragmentation also makes the platform less competitive with more standardized platforms or closed platforms, which 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 ensuring backward compatibility, by utilizing the copyright and patent legal regimes to prevent modifications, by denying access to source code, by using pricing and contracts with third parties to discourage fragmentation, and by 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-platform creator decides 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. The software-platform creator could modify the license and prohibit forking. Doing so, however, would go against core principles of the open source development community, result in a backlash, and likely reduce, perhaps sharply, the supply of programming effort for the platform.

Android’s experience illustrates the issues with soft and hard fragmentation. Google started

http://www.cnet.com/news/apps-on-the-amazon-fire-phone/ and
http://www.huffingtonpost.com/2014/07/22/amazon-fire-phone-review_n_5611573.html. Only 230,000 -240,000 apps are available on Amazon App Store, compared to 1.2 million on Google Play.


44 For example, even for the relatively homogeneous iOS platform, developers have to make sure their applications work on several versions of the operating system and multiple devices including iPhone, iPad, and iPod Touches. See, for example, Tony Bradley, Apple iOS is More Fragmented Than it Seems, FORBES, June 21, 2013, http://www.forbes.com/sites/tonybradley/2013/06/21/apple-ios-is-more-fragmented-than-it-seems/

45 It violates the first freedom of the Free Software Foundation. See http://www.gnu.org/philosophy/free-sw.html
developing a mobile phone operating system after acquiring Android in 2005.\textsuperscript{46} Google organized the Open Handset Alliance to collaborate on marketing the platform. The Android software source code is made available under the Apache Software License, Version 2.0. Google set up the Android Open Source Project to manage the development effort. As is typical when a for-profit company leads an open-source effort, Google hires the core developers and leads the development effort.

Google's approach to software platform development has helped drive positive network effects among users, developers, and hardware makers. That is seen in the figures on users, applications and hardware devices discussed above. This approach has also, however, resulted in fragmentation issues that impose negative externalities on the Android community. There is soft fragmentation. Google releases periodic updates to Android but hardware makers and wireless carriers often do not immediately incorporate the most recent versions in new phones or provide software updates for installed phones. That, together with the fact that people make purchases from different manufacturers at different points in time results in the existence of multiple software version and hardware combinations.

Android software developers have found this fragmentation a challenge. For example, as of August 2014, a research report by OpenSignal pointed out there were 8 versions of Android in use, and that

\begin{quote}
“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.”
\end{quote}

OpenSignal reported that there were 18,796 distinct Android devices in 2014 (43 percent more than in 2013). Each separate device presents an added risk that an application will not work properly. Developers could test their applications on multiple devices. However, according to OpenSignal, the top ten devices only account for 21 percent of all devices in use.

Part of the 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, 8 percent had the next oldest version and only 1 percent had earlier versions.\textsuperscript{48}

Android also faces more serious fragmentation issues that result from handset makers developing their own versions of the Android operating system, which results in their version being

\begin{flushright}
\end{flushright}

\begin{flushright}
\end{flushright}

\begin{flushright}
\textsuperscript{48} The most recent version of iOS is iOS 8, released in September 17, 2014.
\end{flushright}
incompatible with other versions. As I noted above some manufacturers such as Amazon have “forked” Android and developed versions that are not compatible with most Android applications.

C. 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. That can happen in a multitude of ways.

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.\footnote{See, Rob Stafford, Why parents must mind MySpace, NBC NEWS, April 5, 2006, http://www.nbcnews.com/id/11064451/ns/dateline_nbc/t/why-parents-must-mind-myspace/#.VCLuH010yM8} Among other things it attracted a number of pedophiles.\footnote{See Felix Gillette, The Rise and Inglorious Fall of MySpace, BLOOMBERG BUSINESSWEEK, Jun. 27, 2011, http://www.businessweek.com/magazine/content/11_27/b4235053917570.htm} At the same time, MySpace had difficulties matching advertising and content. 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.\footnote{For a general overview of the problems, see Felix Gillette, The Rise and Inglorious Fall of MySpace, BLOOMBERG BUSINESSWEEK, Jun. 27, 2011, http://www.businessweek.com/magazine/content/11_27/b4235053917570.htm}

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.\footnote{Kevin J. Boudreau and Andrei Hagiu, Platform Rules: Multi-Sided Platforms as Regulators in PLATFORMS, MARKETS AND INNOVATION (Edward Elger Publishing, 2009). Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1269966} The successful game console companies such as Sony (for its PlayStation) that followed Atari limited the ability of third parties to publish games for their platforms and imposed quality controls.

D. COMPETITION AMONG AND BETWEEN SOFTWARE PLATFORM ECOSYSTEMS

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, like all multi-sided platforms, they can also differentiate themselves by the pricing structure, which determines the relative participation of the various sides, and 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 also decide whether to differentiate the platform itself by providing multiple versions of the platform with different features.\(^{53}\)

These decisions concerning differentiation result in externalities because, by influencing demand by members of one group for the platform, they affect the demand by members of the other groups of 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. Hardware makers and even application developers can make modifications to the software platform code under standard open source licenses. Those modifications could make some hardware and software incompatible. Moreover, developers could provide alternative and potentially incompatible versions of the software platform. These considerations echo some aspects of the tradeoff between interbrand and intrabrand competition.\(^{54}\)

Brand owners can organize the production and distribution of their products in a variety of ways including working with other firms. A common situation involves a brand owner manufacturing a product and relying on distributors to sell the product to customers. Brand owners can choose the degree of control or production and distribution. They can keep production and distribution tightly under their control or delegate more control to business partners. Tight control may ensure quality and consistency while less control may result in lower prices and perhaps even innovation by partners. Intrabrand restrictions—particularly exclusive territories and price restraints—encourage distributors to invest in service, promotions, and repair facilities.

---

\(^{53}\) Microsoft, for example, released four different versions of Windows XP in 2001: Starter, Home, Professional, and 64-Bit.

In some cases more intrabrand competition could increase the value of the product to customers and thereby enhance interbrand competition. In other cases intrabrand competition could reduce quality and consistency and thereby harm interbrand competition. These considerations apply generally to software platforms. However, unlike most products considered in the literature on interbrand versus intrabrand competition, decisions on the degree of control can result in significant positive or negative externalities. Those externalities can in turn lead to positive or negative feedback effects for participants in the platform ecosystem. For software platforms, fragmentation raises particularly serious concerns over negative externalities that could reduce the value of the platform overall.

IV. 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, based on rules and standards, to harness externalities to maximize the value of their platforms. Perhaps more surprisingly, successful open-source software platforms have developed governance systems as well to deal with positive and negative externalities.

Multi-sided platforms generally face these sorts of externalities and have devised governance regimes for them. It is useful to start with an overview of generally before turning to the specific governance approaches for software platforms.

A. GOVERNANCE OF MULTI-SIDED PLATFORM COMMUNITIES

Communities involve interdependent economic agents who can create positive and negative externalities for one another. Governance systems typically emerge to deal with these externalities. Polities adopt criminal and civil laws that restrict the freedom of action of their residents. People are not supposed to inflict bodily harm, are supposed to follow contracts, and adhere to community standards. Polities have police and judicial systems to enforce these rules. They also adopt policies to provide public goods and promote positive externalities among residents. They fund building roads and operating fire departments, for example, and of course, have the power of taxation to pursue

---

55 There is also a related literature on the anticompetitive use of restrictions to intrabrand competition. In addition to the issues discussed in that literature the analysis of multi-sided platforms such as software platforms would also need to consider the role of externalities. I return to this issue later.

Economic agents sometimes develop institutions spontaneously for governing externalities among them. Elinor Ostrom has identified many situations in which economic agents develop governance systems for common-pool resources. These systems do not rely on the market, assignment of property rights to a single firm, or to the state through regulation.\(^{57}\) In 1435, for example, Spanish farmers adopted rules that assigned rights to water, including how to share water during droughts, maintenance of canals, and imposition of fines. The farmers monitored each other and established a tribunal for hearing disputes.

Multi-sided platforms arise when it is possible to generate positive externalities through the creation of a community. The participants of a multi-sided platform come together on the platform. They are interdependent in a variety of ways. In addition to membership and usage externalities, positive and negative externalities typically flow between members on different sides and on the same side just as there are between members of any community.

Many multi-sided platforms are started by entrepreneurs who identify profit opportunities from creating a community and harnessing externalities. Consider a shopping mall. There are membership externalities in that more retailers benefit shoppers and more shoppers benefit retailers. There are also externalities between stores. An upscale department store may attract customers for an upscale clothing boutique while a discount department store might diminish the appeal of an upscale jewelry store nearby. There are externalities between shoppers as well. Hooligans roaming the mall create negative externalities while fashionable men and women create positive ones. There are also negative externalities between retailers and shoppers as when stores sell shoddy merchandise or shoppers engage in fraud or theft. By governing these externalities the shopping mall owner can maximize the value of the mall and its profits.

Some multi-sided platforms arise when a community comes together to manage common-pool resources similar to those described by Ostrom. Merchants sometimes cooperate to create shared platforms.\(^{58}\) For example, Newbury Street, in Boston, is an eight-block street, well known for its upscale boutiques, salons, and restaurants. In 1972, some of the businesses on the street formed the Newbury Street League.\(^{59}\) The League, which has 300 members, works to “beautify and maintain” Newbury Street. It hires someone to maintain the trees along the street and has improved the street by, for example, getting new lampposts. In addition to marketing and advertising, it sponsors events such as holiday strolls and Christmas tree lighting ceremonies to attract shoppers.\(^{60}\) These policies promote positive externalities between shoppers and merchants.\(^{61}\)

---

57 See, for example, Elinor Ostrom, Governing the Commons: The Evolution of Institutions for Collective Action Chapter 3 (Cambridge University Press, 1990).
59 See http://www.newburystreetleague.org/about-membership2/
61 Polities themselves operate what are in effect multi-sided platforms. Urban planning addresses issues such as providing nice places for people to reside, work, and shop. Singapore, a city-state, has focused its economic development efforts over the years in providing a trading platform for various sorts of businesses. Recommendations by the Services Sub-Committee Working Group on Trading, Developing Singapore into a Global Trade Hub (2002),
Like other communities, multi-sided platforms adopt governance systems to deal with these externalities. Commercial platforms typically do this through contracts with their members and through their ability to enforce their property rights. Shopping malls, for example, have security details that can deal with unruly shoppers and contracts that impose requirements on stores. Cooperative platforms may deal with negative externalities through peer pressure or expelling members.

B. Governance Tools for Dealing with Externalities

Multi-sided platforms adopt a variety of methods to deal with externalities. That includes pricing and design decisions. Three particular aspects of platform governance are relevant for the subsequent discussion of software platforms: “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).

1. Standards

Multi-sided platforms often have “standards” that determine how platform participants interact and that shape expectations concerning those interactions. By reducing transaction costs standards foster positive externalities and reduce negative ones. Standards sometimes involve common formats for interactions between platform participants. Twitter adopted a maximum 140-character statement as the standard method of communicating on its micro-blogging platform. That told message receivers what to expect and message senders what to do. YouTube initially adopted a maximum 10-minute video as the standard for its video platform. Standards also include restrictions on who can join the platform. Sometimes these are explicit. During its early years only college students (determined by whether they had a “.edu” address) could join Facebook. In other cases they involve an “exclusionary vibe” that communicates what kind of participants are welcome, or not welcome on the platform. The advertisements for farmersonly.com strongly suggest that anyone who doesn’t like country living should stay away. In practice platforms set standards in a variety of ways. Shopping malls provide a standard platform through various design decisions. A developer may create a standard place for upscale stores and well-off shoppers to meet by choosing a location near high-income shoppers and renting stores to high-end stores.

In each of these cases the standard potentially restricts some participants. Verbose people may not want to use Twitter, producers of long-form videos for many years could not upload to YouTube, partners at city law firms are not welcome on farmersonly.com, and discount stores

---


62 For the key articles that discuss this concept for multi-sided platforms see Note 4.
cannot get space at an upscale mall. Yet the standard likely increases positive externalities by increasing the density of people who want to connect with each other. Relaxing the standard, to permit more possible interactions on the platform, could increase the number of platform participants all else equal. But it could also increase the transaction cost of finding a match and thereby decrease the value of the platform to these participants. That in turn would discourage people from joining the platform. It seems likely that Twitter and YouTube grew very rapidly in large part because of these decisions to adopt standards.

Multi-sided platform standards can also be viewed as reducing negative externalities between members. Those who want to meet a farmer will incur added costs if they have to sort through urbanites, dentists, and other people they do not care to meet. A person interested in watching a short video on their smartphone may not like finding out that the content they have found is a bandwidth-intensive long-form video.

Standards provide a method for multi-sided platforms to differentiate themselves and thereby compete with other platforms. Micro-blogging services such as Twitter are differentiated from social networks such as Facebook, video sharing sites such as YouTube are differentiated from online-video distributors such as Netflix, farmersonly.com is differentiated from christiancupid.com, and, in the Boston area, the upscale Chestnut Hill Mall is differentiated from the down-market Shoppers World.

2. Rules

As with other communities, multi-sided platforms may also need rules to require or forbid certain behaviors that result in externalities among platform participants.

These rules could involve requirements or prohibitions. Payment card schemes, for example, typically require merchants that enter into contracts to accept their cards to post signs at their stores indicating that. A shopping mall may impose specific design and operational requirements on stores that rent space.63 OpenTable, a restaurant reservation application, requires members to cancel reservations at least thirty minutes in advance.64

The rules could also have prohibitions. Social networks typically restrict members from doing certain things. LinkedIn prohibits members from misrepresenting themselves by, for example, providing a false name, picture or information.65 eBay requires that people who win bids buy the item and, if they have entered auctions for similar items and won more than one, to buy both.66

64 OpenTable Terms of Use, OPENTABLE, http://www.opentable.com/info/agreement.aspx.
3. Enforcement

Standards and rules are self-enforcing for multi-sided platforms as in many communities. Standards and rules become norms of behavior. Economic agents may abide by rules because it is expected of them. They do not want to violate the rules because of reciprocity. They do not want others they deal with to violate the rules either. Perhaps they don’t want to violate the rules because of peer pressure. Economic agents may also abide by them because the standards and rules conform to their own preferences. They may have selected themselves into a platform community that has rules they like. Although technically rules and standards restrict behavior they generally do so in a way that restricts most members from engaging in behavior they would not engage in absent the rule.

Some participants may violate the requirements and prohibitions. Many multi-sided platforms enforce these rules through contracts with their members and exercising their property rights to ban economic agents from their platform properties. Consider OpenTable’s “No Show” policy. Its platform can monitor whether people comply with its requirement that if they are not going to keep the date they cancel their reservation at least 30 minutes in advance. It advises people in its terms and conditions, to which people must agree to use its platform, that it will monitor their cancellation behavior and that they will be forbidden from using the platform if they are “no shows” more than four times in a 12 month period. OpenTable can exclude violators of its policy because it controls access to its platform.

C. Governance of Software Platforms

Software platforms usually adopt governance systems to regulate externalities among the members of their platforms, based on a survey of a diverse group of platforms. Table 3 identifies 15 significant software platforms. They are classified along two dimensions. The vertical columns break these software platforms into those following a pure profit-maximizing firm model that strictly enforces intellectual property rights (“proprietary”); the open source model in which the creator of software platform effectively cedes significant intellectual property rights to third parties (“open-source”); and software platforms where a profit-maximizing firm is the owner of the intellectual property but has chosen to rely to some degree on open-source production and therefore has ceded some intellectual property rights (“hybrid”). The horizontal rows identify several leading hardware platforms for these software platforms: mainframes; personal computers and games consoles; the “Internet”\(^\text{68}\), and mobile devices. These categories cover most of the significant software

\(^{67}\) OpenTable Terms of Use, OPENTABLE, http://www.opentable.com/info/agreement.aspx

\(^{68}\) The “Internet” in this context refers to running a software platform on a web server that is then connected through the Internet (consisting of backbone providers, transit providers, content delivery networks, and Internet Service Providers (ISPs)) to users. The users may access the software platform in this case through their browser or through a mobile application.
platforms. Each cell of the table identifies leading examples.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Software Platforms Examined</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proprietary</td>
</tr>
<tr>
<td>PC and Game Console</td>
<td>Sony PlayStation</td>
</tr>
<tr>
<td></td>
<td>Windows</td>
</tr>
<tr>
<td>Internet</td>
<td>Facebook</td>
</tr>
<tr>
<td></td>
<td>Salesforce</td>
</tr>
<tr>
<td></td>
<td>Tencent</td>
</tr>
<tr>
<td>Mobile device</td>
<td>Apple iOS</td>
</tr>
<tr>
<td></td>
<td>Windows Mobile</td>
</tr>
</tbody>
</table>

Most of these platforms have governance systems that involve standards, rules, and enforcement as shown in Appendix Table A. 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, such as Facebook and the Sony Play Station Game Network, 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. Proprietary platforms typically identify rules and enforce those rules by contracts. They can exclude participants by refusing to enter into contracts with them. Even platforms under contract with participants typically have the contractual rights that enable them to terminate the relationships and seek damages under particular contingencies. 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. Applications and hardware have to pass compatibility tests run by the platform. Only those products that have done so have the right to use trademarks.

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. The Android operating system is a hybrid model. For this case the focus is on dealing with fragmentation and inconsistent experiences for consumers and the tradeoffs that a for-profit company faces in getting the benefits of open source while mitigating the costs from fragmentation. The Bitcoin protocol is an open source software platform for

---

69 One category not include here just for brevity are software platforms that operate across multiple hardware platforms such as Java.
financial services. It is a pure open source model. Lack of centralized control has hindered Bitcoin’s growth and development. Finally, the Windows PC operating system is a pure for-profit model. Microsoft’s standards and rules were the subject of a classic antitrust case and the issue of fragmentation was the subject of some discussion of the remedies for the anticompetitive behavior found by the authorities and courts.

1. The Android Operating System

Android, Inc. was a for-profit company started in 2003 to develop a new mobile operating system. Its founders and development team moved over to Google in 2005, upon its acquisition, and continued their development of the mobile operating system. Google could have adopted a typical for-profit business model for Android. It could have kept the Android code proprietary and used its intellectual property rights over Android in order to prevent others from making any unauthorized modifications. As discussed above, this model would have eliminated fragmentation almost entirely by adopting a more vertically integrated model between software and hardware. Instead, Google decided to adopt a hybrid model. Google announced that it would make Android code available under an open source license in November 2007 and made the Android source code publicly available in October 2008 when the first phones based on Android launched.

Google has established a governance system for guiding the development of the platform. Like most open source endeavors it established a “project” with defined rules and management. The Android Open Source Project (AOSP) coordinates the development of the platform. AOSP releases a standard version of Android. The most recent one is Lollipop 5.0. According to the project description, “Google is responsible for Android product management and the engineering process for the core framework and 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. Google also helped establish the Open Handset Alliance (OHA), which is a group of technology and mobile device manufacturers that work together to promote the Android platform by, for example, sharing intellectual property or contributing to the development of Android.

Google and its partners have increased their efforts to deal with negative externalities in the Android community.

The open source model worked extremely well in the early years. Android provided device makers with a smart phone operating system that was not only superior to the alternatives available

---

70 Android uses the Linux kernel. Regardless of how other parts of Android were licensed the Linus kernel, and any other part relying on open source code, would have to be licensed under the applicable open-source license.


72 Project Roles, ANDROID, [https://source.android.com/source/roles.html](https://source.android.com/source/roles.html)
to them, but also free. Android relied on Java as the programming language for its applications. That enabled it to tap into a large pool of experienced Java programmers. The open-source development model and the relative low learning curve for writing applications attracted developers. Competition between handset makers and the free operating system resulted in the introduction of full-featured smartphones that were priced low relative to Apple’s iPhone. Meanwhile there was a rapid increase in the number of applications being developed for smartphones in general, including for Android. The first Android handset was released in September 2008. The platform grew quickly. By 2010 67.2 million of the 296.6 smartphones sold that year used the Android operating system. That compared to 111.5 million for Symbian and 46.6 million for iOS. Android attracted several large handset manufacturers including Samsung, Motorola, ZTE, HTC, and Huawei. By mid 2010, there were more 70,000 applications in the Android Market.

Since the beginning, Android has had to deal with two forms of fragmentation. The first was ensuring that hardware providers met minimum compatibility standards so that applications developers could be certain that their apps would function on every compliant Android device. To address this, Google developed a set of compatibility standards for hardware makers, which it launched in 2007. The stated purpose of these requirements is to “Provide a consistent application and hardware environment to application developers,” “Enable a consistent application experience for consumers,” “Enable device manufacturers to differentiate while being compatible,” and “Minimize costs and overhead associated with compatibility.”

Google provides various tests (also on a free and open-source basis) that hardware makers can use to ensure that their devices are compatible. Google only allows Google Play (Google’s mobile applications store) and certain Google apps to be preloaded onto devices that are compatible, which helps incentivize OEMs to offer Android compatible devices.

Google also releases an SDK for software developers. By following the guidelines in the kit developers can better assure that their applications work across devices that meet the compatibility standards. Every version of the Android platform has a corresponding version of the SDK exposing the same APIs to developers when they create their applications as will be present on hardware devices implementing that version of the platform. Because hardware manufacturers are free to modify the source code for Android, including the APIs on the device that correspond to those exposed to developers in the SDK, the compatibility guidelines and tests ensure that, for a device certified as compatible for a given version, the same APIs that developers are relying on for that version of the SDK are indeed implemented on the device and offer the same functionality specified to developers.

There is a set of compatibility tests for the application developers as well. The SDKs and the compatibility standards therefore work together to create a platform in which hardware devices can work with applications and applications can work with hardware so long as both sides comply with the standards. It therefore promotes positive externalities. It also prevents negative externalities by preventing hardware makers and application developers from doing certain things that will prevent

---


their respective parts of the platform from working together, leading to a poor user experience.

Google has also faced another form of fragmentation due to the difficulty in persuading hardware makers and carriers to update their devices with the latest version of Android. When Google releases new versions of Android there is no mechanism that requires device makers and wireless carriers to use the most recent version of Android or update previously sold devices with the latest version. In their efforts to differentiate, hardware makers and carriers often customize Android for their devices such that when Android is updated, they have to customize and test the new version before releasing it for their devices (if they do so at all). The proliferation of handsets, releases of different versions of Android, and modifications to Android made by device makers lead to the rapidly increasing fragmentation of the Android platform described above.

One of the ways Google has sought to address the operating system fragmentation problem is by delivering important updates to users through Google Play Services. Google Play Services offers a set of APIs that Google itself can update through the Play store no matter what version of Android the device is running. This demonstrates how platforms can reduce fragmentation not only through rules, but also through innovation.

Google has developed rules and standards that foster positive externalities as well as pursuing its own business interests as the owner of Android. It has licensed the Google Mobile Applications suite (GMS), which includes Google Maps, YouTube, and a few other apps. As part of the license Google requires that the applications appear in particular screens and places on the mobile device. Consumers can therefore form expectations on at least one component that they will get with an Android mobile device “out-of-the-box.” Google has argued that providing the GMS helps Android device makers compete with Apple and Windows phones, which also come preinstalled with software that consumers expect.

Google also operates one of the most popular stores for downloading Android applications. It imposes various quality controls on developers who want to place their applications in the store. It can remove the 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, and engaging in various kinds of malicious and deceptive behavior. In February 2013, Google removed 60,000 low-quality applications from the Play Store.

---

2. The Bitcoin Digital Currency Platform

Bitcoin is a “software-based online payment system” that was created in 2008.\(^\text{80}\) It is based on a software platform that uses a distributed network of servers to process bitcoin transactions, to create more bitcoins, and to 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. Like other open-source software it is registered as a project on SourceForge and open-source developers can contribute to it under the project rules.\(^\text{81}\) As with many open-source software projects there is a non-profit entity that assumes overall management responsibility.\(^\text{82}\)

Despite the media attention Bitcoin has gotten, much of which suggests that Bitcoin is widely used and accepted by merchants, it has failed to reach what is termed “ignition” for multi-sided platforms. Typically, new multi-sided platforms reach critical mass and then go into a period of exponential growth as a result of positive feedback effects; that growth eventually tapers out.\(^\text{83}\) To illustrate this point Figure 1 shows the global transaction trajectory of Bitcoin with the domestic transaction trajectory of mPesa, a successful mobile money platform in Kenya.

---


\(^\text{81}\) Bitcoin: A software-based online payments system, SOURCEFORCE, [http://sourceforge.net/projects/bitcoin/](http://sourceforge.net/projects/bitcoin/).

\(^\text{82}\) BITCOIN FOUNDATION, [https://bitcoinfoundation.org/](https://bitcoinfoundation.org/)

Negative externalities have plagued Bitcoin. The digital currency was originally conceived to handle micro-transactions digitally. The “killer app”, however, was as 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 Bitcoin for money laundering. As a result Bitcoin faced a similar issue as MySpace. These sinister uses of Bitcoin generated positive feedback effects within particular subsets and therefore helped expand the platform. But they also harmed the image of Bitcoin among law-abiding citizens and, in particular, regulators who started reining Bitcoin in.

Bitcoin was successful enough, especially as its exchange rate started to increase, to spawn a number of complementary businesses for the platform. These included exchanges where people bought, sold, and stored bitcoins and vaults where people stored bitcoins. A number of the exchanges and vaults lost bitcoins through either malfeasance by the operators or by cyber-thieves.\(^\text{84}\) The most famous of these was Mt. Gox, which allegedly “lost” about $500 million (calculated at the exchange rate at the time) of bitcoins from, it claimed, hacking.\(^\text{85}\) It filed for bankruptcy. These problems raised the risk of using bitcoins in ways that touched businesses of questionable reputation or viability.

A design flaw in the Bitcoin protocol resulted in another source of negative externalities.\(^\text{86}\) The


\(^\text{86}\) See Ed Felten, *Bitcoin Mining Now Dominated by One Pool*, FREEDOM TO TINKER, Jun 16, 2014, [https://freedom-...
protocol put the processing and creation of bitcoin in the hands of a distributed network of “miners”. Miners compete to solve a mathematical equation that determines whether a new bitcoin transaction is valid and should be added to the public ledger. The winners are in effect entered into a lottery from which they may win a reward of newly minted bitcoins. They may also receive transactions fees. Thus the miners serve two functions—they process transactions for the public ledger and they inject new currency into the system. Winning these contests, however, is a function of mining capacity. In order to share risks miners decided to form coalitions. A mining group that reaches 51 percent of capacity, however, can cause negative externalities. It automatically gets to decide valid transactions and can therefore decide to game the system. It also obtains market power in setting transactions fees. In mid 2014 one of the coalitions, GHash, allegedly crossed the 51 percent threshold. Community pressure persuaded it to reduce its share.

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 bankrupt Mt. Gox. It does not have any apparent levers for regulating the complementary products and services in the ecosystem such as the errant Mt. Gox. It also does not have any power to deal with dominant mining coalitions. 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. In the case of Bitcoin, its creator is anonymous and someone elected under its voting rules leads the foundation.

3. The Windows PC Operating System

Windows was introduced in the mid 1980s and gained widespread adoption in the 1990s. As of 2011, more than 1.2 billion computers globally used Windows and, as of September 2014, more than 90 percent of computers worldwide had Windows installed. Hundreds of computer manufacturers preinstall Windows on their machines although end users can also buy it directly. Many software developers write applications for Windows. There are more than 4 million Windows applications. Microsoft follows a standard for-profit model. It completely controls the

---

89 For sources see, generally, DAVID S. EVANS, ANDREI HAGIU AND RICHARD SCHMALENSEE, INVISIBLE ENGINES: HOW SOFTWARE PLATFORMS DRIVE INNOVATION AND TRANSFORM INDUSTRIES, Chapter 4, (The MIT Press, 2006).
92 As of 2010, Microsoft’s CEO, Steve Ballmer, claimed that were more than 4 million applications for Windows. http://www.cnet.com/news/live-blogging-steve-ballmer/
development, design, and release of the software and carefully protects the source code as a crown jewel. Microsoft does not operate in isolation from the rest of the ecosystem though. As it developed new versions of Windows, with new features, it worked closely with hardware makers and software developers.

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 SDKs that instructed developers on how to develop compatible applications. It did not have any general mechanisms, however, 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. Although the licenses are confidential there is some information available on them as a result of the antitrust case brought by the U.S. Department of Justice and various U.S. states in 1998. 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.” As part of these requirements computer manufacturers were prohibited from launching an alternative user interface, having icons and folders in different sizes and shapes than used by Microsoft, and using a Microsoft feature to display third-party brands. These prohibitions limited the computer makers “flexibility and choices in configuring the PC desktop.” On the other hand it ensured that end users would have a consistent experience.

Like some other platforms Microsoft also gave computer manufacturers rewards for doing certain things that generated positive externalities. These were part of the Market Development Program. “Microsoft implemented MDPs to provide substantial economic incentives for [computer manufacturers] to meet Microsoft-imposed conditions on their configuration of Microsoft’s Windows products.” 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 hundreds of computer manufacturers, manufacturers of peripherals, application developers, Internet content providers, and corporate and personal users. As is well known, some aspects of them were found to exclude competition in violation of the antitrust laws. We will see in the next section, however, that the US courts were very careful to maintain the benefits of the prohibitions and rewards in fashioning remedies for the anticompetitive conduct they found that Microsoft was liable for.

---

94 Id. p. 119
95 Id. p. 227
V. COMPETITION POLICY ISSUES

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 procompetitive. 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.

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 loss from reducing positive externalities or increasing negative ones.

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 supports a presumption that software platform governance systems, and their applications to the participants of the platform, are procompetitive.

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 US 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

1. Abuse of Dominance, Monopolization, and Software Platform Governance

Software platform governance rules involve unilateral non-price practices. They often concern contracts between the software platform and members of the customer groups (or sides) of the platform. Competition policy would ordinarily analyze these contracts as vertical restraints. Under EU law a threshold question for Article 101 and 102 TFEU is whether the software platform is dominant in a relevant antitrust market. Under US law a threshold question for Sherman Section 2 is whether the software platform is a monopoly in a relevant antitrust market or is attempting to monopolize a relevant antitrust market; a threshold question for Sherman Section 1 for vertical restraints is usually whether the firm has significant market power. Two concerns could arise if these 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. 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 alleged that Microsoft engaged in a variety of practices to limit the emergence of software platforms, such as Netscape’s 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 Justice Department complained about involved exercises in market power to protect the Windows monopoly that lacked offsetting efficiency rationales.

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.” It found that these rules were

---

96 It is possible that software platform governance systems could also raise coordinated practice issues. For example, analogous to an issue in resale price maintenance, producers on one side of the platform could coordinate with the platform to impose rules that restrict their competitors. This Article does not consider this issue further.

97 This Article does not consider the determination of the relevant market or the assessment of market power, which is relevant for determining dominance and monopoly.


anticompetitive with one exception. The third rule prohibited OEMs from automatically launching alternative interfaces. The court found that the procompetitive 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 to leverage their dominance in software platforms into the adjacent market for complementary products. That is, the effect of the vertical restraint in on 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. The government also claimed that Microsoft tied Internet Explorer to Windows. Microsoft argued that Internet Explorer was an integral part of the software platform—or part of the standard to use the terminology used above. The District Court agreed under a per se analysis and the D.C. Circuit reversed on the grounds that there were procompetitive benefits from tying for software platforms and that the proper analysis was under the rule of reason.

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

2. Examples of Concerns

Software platforms generate positive network effects when more members join the various sides. New platforms in particular must garner enough members on each side to reach the critical mass for having a viable platform. A platform could adopt rules that have the effect of limiting the ability of members to belong to another platform. A software platform could prohibit application developers or hardware manufacturers from working on competing platforms. Such exclusivity rules could be a pretense for limiting competition from rivals. They could prevent new software platforms from

---

101 The government claimed that Internet Explorer was an application for Windows and that Microsoft’s rules were part of a strategy to monopolize the browser market. The D.C. Circuit rejected this claim because the government had not shown that there was a relevant antitrust market for browsers.
102 United States v. Microsoft 253 F.3d 34 (D.C. Cir. 2001)
getting off the ground by preventing them from securing a critical mass of developers and manufacturers.

Software platforms could also adopt rules that deter platform participants from offering products that reduce the lock-in of other participants to the platform. A software platform could prohibit device makers from having an alternative software platform on the device. That would prevent users from trying a new software platform and its perhaps limited set of applications without giving up all the applications they use with the incumbent software platform. Microsoft, for example, prohibited licensees that installed Windows on computers from pre-installing any other operating systems.104

A software platform might also restrict applications that could reduce lock in. Suppose, for example, that an application provides a substitute for an important feature of the software platform. If it became popular it would make it easier to substitute to other software platforms. A dominant software platform could consider restricting the application that poses this risk and thereby deny it access to a large group of customers that are necessary for the application to secure critical mass.

These practices could also have pro-competitive justifications. They could, for example, be necessary for maintaining critical mass for the benefit of all platform participants or perhaps for dealing with negative externalities.

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. Governance rules that limit fragmentation, if they are successful, tend to narrow the degree of differentiation between variants of the software platform and number of alternative viable versions of the platform. They therefore tend to reduce potential intra-brand competition, which competition policy sometimes frowns on.

104 Microsoft was sued by BE which produced the BeOS operating system. BE claimed that it was unable to license its free operating system to computer makers as a result. See for an overview of the claims. http://www.computerworld.com/article/2586191/technology-law-regulation/be-files-antitrust-suit-against-microsoft.html. Microsoft and BE ultimately settled. See http://news.microsoft.com/2003/09/05/microsoft-corp-and-be-inc-reach-agreement-to-settle-litigation/. 
Governance rules for open-source software platforms are very different from manufacturer restraints on distributors in an important respect, however. 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 related to the sale of the product such as price and service. Governance rules to control fragmentation are equivalent to rules that prohibit distributors from changing the features of the product they are selling.

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.

One could argue that while restrictions may have been efficient when a software platform is young, and needed positive network effects to grow, they are not efficient, and are indeed anticompetitive, when a software platform has become successful. In fact, the likelihood of fragmentation and its costs to the platform ecosystem are likely to become larger as the software platform gains more participants. More platform participants can introduce incompatible code into the software platform or applications. (In some cases they do so actively, an in other cases passively by not upgrading to new versions of the platform.) The number of distinct versions therefore grows exponentially. There is a temporal dimension to this problem as well. Older software platforms, whether under an open-source or proprietary license, have more versions. Therefore the fragmentation problem can become more severe over time.

That has been the case for Android. According to OpenSignal the number of distinct Android devices increased by 370 percent between 2012 and 2014.\textsuperscript{105} As the number of participants on the platform become greater the opportunities for encountering incompatibilities becomes greater as well. Developers have to write for more versions of Android or avoid including more advanced

functionality to be certain that their applications work for all users. Users face increased odds that applications they download will not be fully compatible with their device.

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. 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 was 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 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 burden of showing that the practice provided 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, and open-source licensing model, or a hybrid model of those two.

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

---

106 This three-step process for platform governance systems was proposed in David S. Evans, “Governing Bad Behavior,” op cit., pp. 1247ff.

107 The competition policies of most jurisdictions are based on the EU approach, the US approach, or a blend of those two approaches to antitrust.
evaluate complaints and using their prosecutorial discretion in determining whether a complaint against a governance practice merits close attention.

VI. CONCLUSIONS

Software platforms now anchor vast global communities of personal, corporate and government users; device and peripheral manufacturers; businesses built on applications; content providers; advertisers; and more. They are central to much economic and personal activity. Substantial global industries now depend on these platforms.

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.

Mobile phone operating systems, for example, are central to a vast ecosystem that enables people to obtain various products and services through mobile devices. It includes device manufacturers, chip and other component manufacturers, wireless carriers, application developers, advertisers, and content providers. The application developers are creating massive global businesses on top of these operating systems. Uber’s mobile application, to take one case, is revolutionizing urban transportation.

The value of these software platforms, and their ability to support large communities, depends on the ability of the platform to promote positive externalities and reduce negative externalities. They 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 anti-competitive 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.
## APPENDIX

### Table A

**Examples of Software Platform Governance Systems**

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Standards</th>
<th>Rules</th>
<th>Enforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>The Android Open Source Project releases a standard version of Android. Google also has a set of standards that correspond to a &quot;compatible version&quot; of hardware devices. It also has a SDK and software development guidelines for producing compatible software.</td>
<td>Device makers must meet the compatibility tests in order to market their devices as Android Compatible. Applications must meet certain quality guidelines to be part of the Google Play store.</td>
<td>Google can remove from Google Play 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, and engaging in various kinds of malicious and deceptive behavior.</td>
</tr>
<tr>
<td>Windows</td>
<td>Microsoft releases a standard version of Windows that cannot be changed and a defined set of APIs that can be used. Hardware developers must meet certain standards for having a Windows compatible machine. The SDK imposes standard development guidelines for applications.</td>
<td>Microsoft licenses imposes various requirements on hardware maker licenses such as not having an alternative boot sequence. Many other restrictions were found unlawful and were modified as a result of a settlement with the government.</td>
<td>Microsoft provides rewards in the form of marketing allowances for doing various things that benefit the platform. It also provides information on compatibility that constrains manufacturers from deviating.</td>
</tr>
<tr>
<td>Facebook</td>
<td>Facebook has a highly standardized environment for providing information for people and businesses. The design of the platform provides precise methods and formats for people to communicate with each other. It also provides precise formats for displaying advertising, and for running games. There are other standards such as people must be real.</td>
<td>Facebook has detailed rules for every different type of participant in the platforms. These rules describe what they must or must not do. The include rules for people and businesses that have Facebook pages, developers who write applications, and advertisers.</td>
<td>Facebook can and has excluded anyone who violates the rules from the platform. That includes dropping pages of people who use fake names or engage in profanity, applications that do not meet its requirements, and advertisements.</td>
</tr>
<tr>
<td>Apple iOS</td>
<td>Apple provides a standard hardware environment over a small number of devices that all App developers and users can use. Apple also provides a standard software development environment using Apple's SDK and Xcode 6.</td>
<td>There are specific rules and regulations for applications including size and streaming speed, as well as basic community standard type rules. That policy prohibits applications from a variety of activities including making modifications to the user's device, various kinds of malicious and deceptive behavior, and other restrictions involving: Game Center, iAds, Trademarks, Media Content, User interfaces, Purchase and currencies, Scraping and aggregation, Damage of device, Personal attack, Violence, Privacy, Pornography, Religion, Culture, Lotteries, Charitics, Contributions, and Legal requirements.</td>
<td>Apple must approve applications before they can be listed in the Apple Store. It can remove from App Store applications that violate the terms of Apples's Developer Program Policy.</td>
</tr>
<tr>
<td>Platform</td>
<td>Description</td>
<td>Developer Agreements/Standards</td>
<td>Rules/Contract Enforcement</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Linux</td>
<td>Linux provides a standard distribution with APIs that application developers and hardware makers can use. There are compliance test suites for applications for Linux.</td>
<td>There are rules for how different versions of the software platform itself are released. There are also trademark rules that limit the use of the Linux trademark including to applications that have passed compliance testing. Microsoft's enters into contracts with phone manufacturers. These likely impose constraints on those manufactures but he contracts are confidential.</td>
<td>Linus uses tough talk and shaming to discipline people in the software platform development community. Noncomplying software is not allowed to benefit from using certain Linux trademarks and the Linux foundation can sue for trademark infringement for those who do.</td>
</tr>
<tr>
<td>Windows Mobile</td>
<td>SDK imposes standard methods of development for Windows mobile apps. There are also requirements for hardware developers, such as the type of processor they need to use, how the device is powered, what kind of camera should have and also about audio requirements.</td>
<td>There is App Development Agreement for Windows Store. Developers must consider this Agreement regarding: submission of apps, content of the apps, evaluation and testing, certification of the apps, modification or discontinuance of apps, removal policies, promotion, etc.</td>
<td>Microsoft can restrict publishing the Windows mobile apps, based on: Functionality (apps that crash will be rejected), Metadata (apps that mention certain names will be rejected), Location (apps that do not notify the user will be rejected), and other restrictions involving: Trademarks, Media Content, User interfaces, Purchase and currencies, Scraping and aggregation, Damage of device, Personal attack, Violence, Privacy, Pornography, Religion, Culture, Lotteries, Charities, Contributions, and Legal requirements. Microsoft can file breach of contract claims against hardware manufacturers that don't comply with contracts and it can choose who it enters contracts with.</td>
</tr>
<tr>
<td>Bitcoin</td>
<td>Bitcoin is a &quot;protocol&quot; that provides a standard framework for digital money transmission using the blockchain which provides standards for sending and receiving &quot;bitcoins&quot; and standards for compensating &quot;miners&quot; of bitcoin who process transactions and create bitcoins.</td>
<td>The rules are basically coincident with the standards in that they are hardwired into the software platform and deviating from them prevents a transaction. However, beyond that there is a relatively weak central control—the Bitcoin Foundation—that doesn't have strict rules or an enforcement mechanism.</td>
<td>The rules are basically coincident with the standards in that they are hardwired into the software platform and deviating from them prevents a transaction. However, beyond that there is a relatively weak central control—the Bitcoin Foundation—that doesn't have strict rules or an enforcement mechanism.</td>
</tr>
<tr>
<td>Sony PlayStation</td>
<td>The PlayStation 3 uses the XrossMediaBar (XMB) as its graphical user interface. This provides a standard development platform for games. Developers must be registered with Sony, and they have to follow PSM SDK development platform.</td>
<td>Rules are provided in private and confidential contracts with game developers who pay Sony royalties. Sony's PlayStation Network also has rules for participants and can ban participants for violating rules. There is a detailed Community Code of Conduct.</td>
<td>Sony approves applications so it can decide who to admit to the development program and it can approve the resulting games. It can exclude game publishers from using its platform for violating contracts. It also has a &quot;Community Code of Conduct&quot; that restricts behavior in various ways violation of which can lead to account suspension.</td>
</tr>
<tr>
<td>Ripple</td>
<td>Provides standard platform and rules of money transmission for users. Also provides a universal protocol for money for developers.</td>
<td>No evidence of detailed rules for developers. However, Ripple enters into contracts with various partners including financial institutions. Those contracts likely impose obligations and provide terms for ending relationship.</td>
<td>Contracts with partners likely provides usual contract enforcement rights.</td>
</tr>
<tr>
<td>Firefox</td>
<td>The Firefox developers must use Aurora—the preview build of Firefox, where they can get the latest features before they are released. Developer tools that must be used are: storage inspector, performance tool, frame switcher, coslole-tablet support, a philosophy, page inspector, extension API.</td>
<td>There are rules concerning what developer should do for add ons that will pass the review process and a list of add-ons that are not permitted.</td>
<td>Add-ons written for Firefox are regulated by a review process and a set of policies. All publicly listed add-ons are reviewed by a team of editors before being released into the Mozilla gallery.</td>
</tr>
<tr>
<td>Platform</td>
<td>Description</td>
<td>Contracts/Requirements</td>
<td>Notes</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OpenStack</td>
<td>The developer must interact with an OpenStack cloud using pre-defined set of tools. The tool he chooses depends on his use case in the cloud. These tools are: SDKs (Java, Ruby, Python, .NET, PHP), CLI (Command Line Interfaces) and API (Application programming interfaces). The extensive testing infrastructure of OpenStack relies on Jenkins, tied to the review process.</td>
<td>There are contracts for commercial use and usage guidelines for community use of badges and logos. The contracts for commercial use include compatibility requirements.</td>
<td>The contracts for commercial use can forbid the use of badges and logos and association with open stack for applications that violate rules including don't meet compatibility guidelines.</td>
</tr>
<tr>
<td>Tizen</td>
<td>Provides application development tools based on JavaScript, additionally to an SDK for developers that allows them to use HTML5 and related web technologies.</td>
<td>Tizen applications must not (i) breach any applicable laws, regulations or generally accepted practices or guidelines in the applicable jurisdictions; (ii) contain any material, component or code which could damage, destroy, unduly burden or unreasonably affect software, firmware, hardware, data, systems, services, or networks; or (iii) disable, hack or otherwise interfere with any authentication, content protection, digital signing, digital rights management, security or verification mechanisms implemented in or by the Tizen Certified platform.</td>
<td>Tizen has the ability to enforce behavior by retaining control of which applications are approved on the one hand, and giving rewards in different Challenges they host on the other hand.</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>Ubuntu provides and standard software environment for various devices including desktop and mobile and cloud. The developers must use Ubuntu App Platform, which consists of Ubuntu SDK, QML and HTML5. Ubuntu also has standards for hardware.</td>
<td>The SDK provides rules for developing applications. The review process has implicit rules since applications can be rejected if they aren't compatible, use profanity, etc. There is a certification program for hardware.</td>
<td>Apps that don't meet quality standards can be rejected. Hardware that doesn't meet certification requirements can be rejected.</td>
</tr>
<tr>
<td>Salesforce</td>
<td>Provides a standard CRM software platform.</td>
<td>Contracts with developers and other members of the ecosystem</td>
<td>Based on contracts</td>
</tr>
<tr>
<td>Tencent</td>
<td>Developers have to use Tencent API, which is based on protocols XML, JSON, REST. Other related APIs are: Crowdfunder (projects related to community, charity, environment), LakeBTC (market data thicker), Huobi (Chinese bitcoin).</td>
<td>Developers need to pass the developer qualification review process, which requires photocopies of firm registration documents or national ID card/passport for individual developers.</td>
<td>Tencent can bounce or exclude applications from its platform.</td>
</tr>
</tbody>
</table>
Readers with comments should address them to:

Lecturer David S. Evans
devans@globaleconomicsgroup.com
For a listing of papers 1–600 please go to Working Papers at
http://www.law.uchicago.edu/Lawecon/index.html

602. Saul Levmore, Harmonization, Preferences, and the Calculus of Consent in Commercial and Other
Law, June 2012
603. David S. Evans, Excessive Litigation by Business Users of Free Platform Services, June 2012
604. Ariel Porat, Mistake under the Common European Sales Law, June 2012
608. Lior Jacob Strahilevitz, Absolute Preferences and Relative Preferences in Property Law, July 2012
611. Joseph Isenbergh, Cliff Schmiff, August 2012
613. M. Todd Henderson, Voice versus Exit in Health Care Policy, October 2012
615. William H. J. Hubbard, Another Look at the Eurobarometer Surveys, October 2012
616. Lee Anne Fennell, Resource Access Costs, October 2012
617. Ariel Porat, Negligence Liability for Non-Negligent Behavior, November 2012
618. William A. Birdthistle and M. Todd Henderson, Becoming the Fifth Branch, November 2012
620. Rosa M. Abrantes-Metz and David S. Evans, Replacing the LIBOR with a Transparent and Reliable Index of Interbank Borrowing: Comments on the Wheatley Review of LIBOR Initial Discussion Paper, November 2012
621. Reid Thompson and David Weisbach, Attributes of Ownership, November 2012
626. David S. Evans, Economics of Vertical Restraints for Multi-Sided Platforms, January 2013
632. Adam B. Cox and Thomas J. Miles, Policing Immigration, February 2013
633. Anup Malani and Jonathan S. Masur, Raising the Stakes in Patent Cases, February 2013
637. Lior Jacob Strahilevitz, Toward a Positive Theory of Privacy Law, March 2013
639. Lisa Bernstein, Merchant Law in a Modern Economy, April 2013
640. Omri Ben-Shahar, Regulation through Boilerplate: An Apologia, April 2013
641. Anthony J. Casey and Andres Sawicki, Copyright in Teams, May 2013
643. Eric A. Posner and E. Glen Weyl, Quadratic Vote Buying as Efficient Corporate Governance, May 2013
646. Stephen M. Bainbridge and M. Todd Henderson, Boards-R-Us: Reconceptualizing Corporate Boards, July 2013
647. Mary Anne Case, Is There a Lingua Franca for the American Legal Academy? July 2013
651. Maciej H. Kotowski, David A. Weisbach, and Richard J. Zeckhauser, Audits as Signals, August 2013
652. Elisabeth J. Moyer, Michael D. Woolley, Michael J. Glotter, and David A. Weisbach, Climate Impacts on Economic Growth as Drivers of Uncertainty in the Social Cost of Carbon, August 2013
656. Evidentiary Privileges in International Arbitration, Richard M. Mosk and Tom Ginsburg, October 2013
658. The Impact of the U.S. Debit Card Interchange Fee Regulation on Consumer Welfare: An Event Study Analysis, David S. Evans, Howard Chang, and Steven Joyce, October 2013
659. Lee Anne Fennell, Just Enough, October 2013
662. Have Inter-Judge Sentencing Disparities Increased in an Advisory Guidelines Regime? Evidence from Booker, Crystal S. Yang, March 2014
665. Lee Anne Fennell and Eduardo M. Peñañver, Exactions Creep, December 2013
666. Lee Anne Fennell, Forcings, December 2013