REVISTA CHILENA DE DERECHO Y TECNOLOGÍA

SEGUNDO SEMESTRE 2015 VOL. 4 NÚM. 2

FACULTAD DE
DERECHO
UNIVERSIDAD DE CHILE
CENTRO DE ESTUDIOS EN DERECHO INFORMÁTICO

La *Revista Chilena de Derecho y Tecnología* es una publicación semestral del Centro de Estudios en Derecho Informático de la Facultad de Derecho de la Universidad de Chile que tiene por objeto difundir en la comunidad jurídica nacional, regional e internacional, el conocimiento científico relevante y necesario para analizar y comprender los alcances y efectos que el desarrollo tecnológico y cultural han producido en la sociedad, especialmente su impacto en las ciencias jurídicas y sociales.

Revista Chilena de Derecho y Tecnología
Rev. chil. derecho tecnol. (impr.)
Centro de Estudios en Derecho Informático
Facultad de Derecho · Universidad de Chile
Pío Nono núm. 1, 4.º piso, Providencia
Santiago de Chile

+56 2 29785263
rchdt@derecho.uchile.cl
http://www.cedi.uchile.cl
http://twitter.com/rchdt

ISSN 0719-2584

La *Revista Chilena de Derecho y Tecnología* es publicada en formatos electrónicos (pdf, epub y mobi) disponibles para descarga en la página web <http://www.rchdt.uchile.cl/>.

Una guía para la presentación de manuscritos a la *Revista Chilena de Derecho y Tecnología* está disponible en el siguiente enlace: <http://www.revistas.uchile.cl/index.php/rchdt/about/submissions#authorguidelines>.

REVISTA CHILENA DE DERECHO Y TECNOLOGÍA
SEGUNDO SEMESTRE 2015 VOL. 4 NÚM. 2
Rev. chil. derecho tecnol. (impr.)

REVISTA CHILENA DE DERECHO Y TECNOLOGÍA
CENTRO DE ESTUDIOS EN DERECHO INFORMÁTICO • UNIVERSIDAD DE CHILE
ISSN 0719-2584 • VOL. 4 NÚM. 2 (2015) • PÁGS. 179-213 • DOI 10.5354/0719-2584.2015.37361

Oracle versus Google: Protección de derecho de autor sobre elementos funcionales de programas computacionales que ponen en riesgo la interoperabilidad y la innovación

Oracle versus Google: Copyright protection on functional elements of softwares that puts at risk interoperability and innovation

MARÍA PAZ CANALES LOEBEL
Berkman Center for Internet & Society, Harvard University, Estados Unidos

RESUMEN El caso *Oracle v. Google* proporciona una buena oportunidad de reflexión sobre el alcance de la protección de los derechos de autor sobre programas computacionales en tanto obras funcionales. Este caso se refiere al uso por Google de las interfaces de programación de aplicaciones (API), desarrolladas por Sun (Oracle hoy) como parte del lenguaje de programación Java, creado con el propósito de estandarizar la comunicación entre diferentes capas de programas computacionales. Para desarrollar Android en forma compatible con los programas computacionales programados en lenguaje Java, Google tomó los encabezados y la secuencia, estructura y organización (SSO) de treinta y siete API de Java. El caso tiene una enorme repercusión en el entendimiento del ámbito de protección del derecho de autor sobre programas computacionales que son obras funcionales por naturaleza. Una revisión de la realidad económica en el desarrollo de programas computacionales muestra

que existen importantes economías de red que se intentan capturar a través de la interoperabilidad mediante el desarrollo de estándares. Esta realidad no debe ser ignorada a la hora de diseñar la política de derecho de autor sobre protección de programas computacionales, de tal forma de prevenir un daño a la innovación futura. Si el derecho de autor falla en esta tarea, la normativa de libre competencia podría ser capaz de proporcionar algunas soluciones que se exploran aquí.

PALABRAS CLAVE Programas computacionales, API, interoperabilidad, derecho de autor, libre competencia.

ABSTRACT Oracle v. Google case provides a good opportunity of reflection about the scope of software copyright protection as functional works. This case is about Google's using of Application Programming Interfaces (APIs), developed by Sun (Oracle today) as part of Java programming language created with the purpose to standardizer communication between different layers of software. To develop Android in compatible way with Java language programmed softwares, Google took the hedgers and the sequence, structure and organization (SSO) of thirty-seven Java APIs. The case has a huge repercussion in the understanding of the scope of copyright protection over software that are by nature functional works. A review of the economic reality in software programing shows that there are important networks economies that software programing tries to achieve through interoperability by the development of standards. This reality should not be ignored at the time to design copyright policy around software protection, in order to prevent the damage to future innovation. If copyright fails in this task, antitrust law could be able to provide some remedies that are explored here.

KEYWORDS Software, API, interoperability, copyright, antitrust.

INTRODUCCIÓN

El caso *Oracle v. Google* ha generado gran interés en la comunidad internacional por las repercusiones que su decisión puede tener en la definición del alcance de la protección de derecho de autor sobre los elementos constitutivos de un programa computacional. Más allá del

caso concreto —que enfrenta a dos gigantes de la provisión de servicios computacionales a nivel internacional—, la materia en discusión atrae el interés de innovadores, programadores y la comunidad académica ocupada de la innovación tecnológica, pues se presenta como una oportunidad para definir los márgenes de la protección de derecho de autor sobre programas computacionales, que sean consistentes con el incentivo a la innovación y la competencia en méritos de productos compatibles con distintas plataformas.[1]

La pregunta acerca de la protección bajo derecho de autor del código de un programa computacional —ya sea fuente u objeto— ha sido resuelta por la ley en la afirmativa en gran parte de las jurisdicciones a nivel internacional —entre ellas en los Estados Unidos, así como en Chile—, concediendo dicha protección en cuanto el código constituye una expresión original. Sin embargo, el caso *Oracle v. Google* nos enfrenta a un análisis más minucioso acerca de qué es aquello que exactamente se encuentra protegido al proteger un programa computacional, y, a la vez, a la pregunta de política pública acerca de la conveniencia de extender dicha protección a elementos de interoperabilidad entre programas computacionales.

Con un último capítulo en este caso pendiente de escribirse en los tribunales norteamericanos,[2] la reflexión acerca del alcance de la protección de derecho de autor sobre los elementos constitutivos de un programa computacional resulta relevante y necesaria para definir el contorno de la aplicación de niveles de protección adecuados, que sean consistentes con incentivar a creadores e inversionistas en el desarrollo de nuevos y mejores programas computacionales, capaces de interoperar con distintas plataformas.

Una excesiva protección que bloquee el uso de elementos de interope-

1. Para una introducción general a los objetivos de política pública usualmente comprendidos en un régimen de protección de derechos de autor pueden consultarse Gordon y Bone (2000) y Van Houweling (2005).

2. El Circuito Federal al revocar el veredicto de primera instancia como resultado de la apelación lo reenvió para su decisión de fondo en materia de *fair use*. Luego de la sentencia de apelación, las partes solicitaron la intervención de la Corte Suprema a través del recurso procesal correspondiente (*Certiorari Petition*). La Corte Suprema declinó oír el caso el 29 de junio de 2015, con lo cual la corte de primera instancia deberá pronunciarse en materia de *fair use*.

rabilidad puede poner un obstáculo considerable a la innovación futura, encareciendo en forma injustificada el acceso a elementos entendidos hasta ahora como estándares técnicos y elementos constituyentes susceptibles de ser libremente utilizados por la comunidad internacional. El uso de elementos de interoperabilidad en programas de computación permite a los innovadores capitalizar la inversión en aprendizaje desarrollada previamente por los usuarios, de manera de proporcionarles la alternativa de adoptar nuevos productos sin enfrentarlos a costos que desincentiven su adopción por razones no relacionadas con el mérito intrínseco del nuevo producto.

En las siguientes líneas veremos cómo en el desarrollo del caso *Oracle v. Google* el análisis anterior se encuentra fuertemente presente. Revisaremos las lecciones que este caso nos deja acerca de los límites de la protección de derecho de autor sobre programas computacionales, la forma en la cual la legislación vigente en Chile puede ser interpretada en consistencia con una política de fomento a la innovación en el desarrollo de programas computacionales, y concluiremos con una breve reflexión acerca de las consecuencias competitivas de la adopción de estándares propietarios sobre elementos funcionales de programas computacionales.

EL CASO ORACLE V. GOOGLE

En el año 2007, Google estrenó su sistema operativo para dispositivos móviles denominado Android. Éste fue desarrollado usando las interfaces de programación de aplicación (API, por su nombre en inglés) que forman parte del leguaje de programación Java, creado por Sun Microsystems en los años noventa,[3] y ofrecido a la comunidad internacional de programadores computacionales como una plataforma abierta para facilitar la interoperabilidad en la programación computacional,

3. «La estrategia primaria de Sun durante el fin de los noventa fue establecer nuevas plataformas independientes de Microsoft que pudieran limitar (si no reducir) el control de Microsoft sobre estándares de la industria. La mayoría de los esfuerzos de Sun se dirigieron al establecimiento de Java como una nueva plataforma con set común de API disponible para un amplio rango de sistemas computacionales, bajo el eslogan 'escrito una vez, corre en cualquier sitio'» (West, 2003: 1275). En lo sucesivo, a menos que se indique lo contrario, las citas corresponden a nuestra traducción.

permitiendo que un mismo código pudiese ejecutarse sin problemas independiente del sistema operativo o plataforma.[4]

En términos generales, las API permiten que el código particular de un programa sea capaz de comunicarse con el sistema operativo de tal forma que el programa pueda ejecutarse en el terminal respectivo, que es capaz de reconocer las instrucciones y transmitirlas en forma legible hacia el usuario que interactúa con la aplicación o programa. Las API identifican un conjunto de funciones de uso general, por ejemplo, una operación matemática, o una instrucción de imprimir o copiar una información en pantalla. Al utilizar las API de Java los programadores incorporan una lista preexistente de tales funcionalidades incurriendo sólo en el trabajo de programar el código que las implementa (las órdenes contenidas en el programa para que la función se ejecute), pero la instrucción o identificación de la función y la forma organización en que dicha instrucción se encuentra son reconocibles para cualquier sistema operativo que opere con lenguaje Java, con independencia de las diferencias que puedan existir entre las instrucciones de implementación de cada programa.

Así, las API desarrolladas para el lenguaje Java facilitan una comunicación universal entre programas y sistemas operativos. Estamos hablando, en definitiva, de un estándar de interoperabilidad[5] que permite tanto a programadores como a usuarios evadir los costos de cambio entre sistemas operativos o plataformas, permitiendo la selección más eficiente de programas ajustados a sus necesidades que pueden correr indistintamente en cualquier plataforma o sistema operativo.

En 2010, Sun Microsystems fue adquirida por Oracle, abarcando con

4. «Cuando Java fue introducido por primera vez en 1996, las API incluían ocho paquetes de programas preescritos. Al menos tres de tales paquetes eran paquetes 'centrales', de acuerdo a Sun, fundamentales para poder usar el lenguaje Java. Estos paquetes eran java.lang, java.io y java.util. Como una cuestión práctica, cualquiera era libre para usar el lenguaje en sí mismo (como Oracle lo reconoce) y debía usar estos tres paquetes centrales en orden a hacer un uso productivo del lenguaje. Contrario a lo afirmado por Oracle, no existe una clara línea divisoria entre el lenguaje y las API», *Oracle America, Inc. v. Google Inc.*, 872 F.Supp.2d 974 (2012), párrafo 982, disponible en <http://bit.ly/1SPXZSU>.

5. Lemley define en forma amplia *estándar* como «cualquier conjunto de especificaciones técnicas que provee o está dirigido a proveer un diseño común para un producto o proceso» (2002: 1896).

ello la adquisición de derechos sobre el lenguaje de programación Java. Luego de dicha adquisición —e ignorando completamente la activa gestión desarrollada en el pasado por Sun Microsystems por elevar el lenguaje Java a la categoría de estándar internacional[6]—, Oracle demandó a Google por la supuesta infracción de Android sobre sus entendidos derechos de propiedad intelectual (patentes y derechos de autor) sobre 37 paquetes de API de Java incorporadas en Android. La infracción de las patentes alegadas por Oracle fue rápidamente desechada, sin embargo, las compañías entraron a una sofisticada batalla legal por demostrar el ámbito de cobertura de la protección de derecho de autor sobre las API en disputa.

En mayo de 2012, la Corte del Distrito Norte de California[7] resolvió en primera instancia que Android no infringía los derechos de autor de Oracle por el hecho de utilizar las API de Java, ya que Android cuenta con su propio y original código de implementación, y aquella parte de las API de Java tomada por Google eran sólo las etiquetas que permiten identificar las funciones de las API (declaraciones) y su forma de organización (estructura, secuencia y organización o SSO) para permitir interoperabilidad. En efecto, Google había replicado los nombres, las funciones y la forma de organización de 37 paquetes de API de Java, pero había creado su propio código para ejecutar tales instrucciones.

El razonamiento de la corte de primera instancia alcanzó correctamente la comprensión de la función del derecho de autor como una herramienta destinada a la protección de expresiones originales, y, en consecuencia, consideró que no existía infracción por parte de Google al haber creado Android usando los nombres cortos genéricos (declaraciones) e implementando funcionalidades equivalentes a las desarrolladas por las API de Java, pero escribiendo para ello su propio código fuente. La corte incluso fue un paso más allá al declarar que «cuando existe sólo una forma de expresar una idea o función, entonces cada uno es libre de hacerlo y nadie puede monopolizar tal expresión».[8]

6. Conforme reportan destacados académicos, Sun Microsystems realizó previamente esfuerzos para lograr que la International Organization for Standardization (ISO) certificara Java como un estándar de programación a nivel internacional (cf. Lemley y McGowan, 1998a: 715; Sliwa, 1999).

7. *Oracle America, Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (2012).

8. *Oracle America, Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (2012), párrafo 976.

Para ilustrar lo extrema que resulta la pretensión de protección de los títulos descriptivos de las funciones y SSO de las API de Java, ello sería equivalente a intentar proteger como obra original la secuencia de organización de un manual o novela que contemple: prefacio, introducción, desarrollo en capítulos, conclusión y prólogo; o un menú de restaurant que contemple: aperitivo, entrada, sopa, plato principal, postre y bajativo. No cabe duda de que no existe expresión creativa en la selección de tales términos o en su organización, ya que corresponden a prácticas estandarizadas por la convención social. Cualquier eventual creatividad podría estar dada exclusivamente por las expresiones utilizadas para la implementación de la función identificada a través del código de programación. A juicio de la corte de primera instancia, Android no infringía los derechos de autor de Oracle al contar con su código fuente original de implementación de las mismas funciones y organización presentes en Java.

La corte de primera instancia sustentó su decisión en la jurisprudencia de la aplicación de la sección 102 (b) de la Copyright Act, conforme a la cual «en ningún caso la protección de derecho de autor se extiende a las ideas, procedimientos, procesos, sistemas, métodos de operación, conceptos, principios, o descubrimientos, sin importar la forma en la cual se describan, expliquen, ilustren, o se incorporen en una obra», que consagra a juicio de la doctrina más extendida la dicotomía expresión/idea que fija los límites del alcance de la protección de derecho de autor. La interpretación de la corte de primera instancia a la luz de la jurisprudencia de aplicación de dicha disposición a la SSO de programas computacionales, la llevó a concluir que la SSO de las API de Java es un método de operación en el sentido descrito en la sección 102 (b) que resulta funcional para la interoperabilidad de los programas, y, por tanto, no es susceptible de protección bajo derecho de autor. Para la corte, «el método de especificación [título de la API tomado por Google de Java] es la idea. El método de implementación [código fuente de implementación desarrollado por Google] es la expresión. Nadie puede monopolizar la idea».[9] A juicio de la corte, si alguna protección correspondería a tales elementos, esa protección debiera encontrarse en el ámbito de las patentes destinada a la protección de las invenciones «útiles», y que por tanto

9. *Oracle America, Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (2012), párrafo 998.

cuentan con una exigencia de «novedad» más alta que aquella requerida por la protección de derecho de autor. Como hemos adelantado, la protección por las patentes alegadas por Oracle fue rápidamente descartada por el tribunal.[10]

El análisis de la corte fue complementado con la aplicación de la doctrina de la fusión (*merger doctrine*), conforme a la cual la protección de derecho de autor no se extiende a aquellas ideas que se encuentran fusionadas con su forma de expresión, de tal forma que no pueden expresarse eficientemente de otra manera. En este caso, las API serían la única forma de expresión eficiente para alcanzar la interoperabilidad que representa el centro de su funcionalidad. La corte concluyó que para generar la funcionalidad deseada se requiere un nivel de precisión, al llamar la función, que sólo podía ser alcanzado utilizando los mismos nombres o títulos usados por las API de Java.[11]

La decisión de primera instancia fue apelada, oportunidad en la cual la Corte del Circuito Federal —competente en segunda instancia en materia de patentes que fueron inicialmente invocadas en la demanda— revocó la decisión de primera instancia en lo relativo a la imposibilidad de proteger las API bajo derecho de autor[12] —corroborando su ya tristemente célebre reputación de expandir la interpretación del alcance de la protección de derechos de propiedad intelectual, que ha requerido una reciente activa intervención de la Corte Suprema de los Estados Unidos para enmendar sus decisiones[13]—, y reenvió el asunto a la corte de pri-

10. No resulta sorprendente que la pretensión de Oracle haya fallado a la hora de demostrar la cobertura de la patente sobre las declaraciones de las API de Java y su SSO, precisamente por cuanto la exigencia de novedad en la patentes, y no tan solo de originalidad, establece un baremo alto a la protección de invenciones útiles.

11. «La declaración requiere de precisión. Significativamente, cuando existe una sola forma de escribirla, la doctrina de la fusión impide a cualquiera reclamar un derecho de autor exclusivo sobre esa expresión. Por lo tanto, no puede haber violación de derecho de autor por usar idénticas declaraciones.» *Oracle America, Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (2012), párrafo 998.

12. *Oracle America, Inc. v. Google Inc.*, 750 F. 3d 1339 (2014), disponible en <http://bit.ly/1OsRwIo>.

13. Por vía ejemplar, el reciente caso de patente sobre software *Alice Corp. v. CLS Bank International*, 573 U.S., 134 S. Ct. 2347 (2014), en el cual la Corte Suprema revocó la decisión del Circuito Federal de conceder protección en sede de patente a un

mera instancia para la determinación acerca de si el uso de las API de Java efectuado por Google satisface o no los requisitos para ser considerado un uso justo (*fair use*), esto es, un uso exceptuado de autorización por parte del titular de derechos de autor.

La decisión de la corte de apelación se basó en la consideración de una línea de jurisprudencia desarrollada en Estados Unidos conforme a la cual la protección de derecho de autor sobre programas computacionales abarca tanto elementos literales como no literales del programa respectivo. Entre los primeros se encuentra la protección del código fuente y del código objeto del programa. Entre los segundos, se incluirían la SSO y las API «expresivas», esto es, dotadas de suficiente originalidad. Un leve ajuste en la estrategia de defensa desarrollada por Oracle en segunda instancia —que se dirigió a subrayar la copia literal de Google de las declaraciones o nombres de las API de Java, que fue considerada entonces una copia de parte del código (independiente de que la implementación hubiese sido desarrollada por Google a través de su propio código fuente)— selló el entendimiento de la corte de segunda instancia en cuanto a que Google habría copiado tanto elementos literales (declaraciones) como no literales (SSO) de 37 paquetes de API de Java.

La corte de apelación estimó que el diseño de los paquetes de API de Java era original y, por lo tanto, susceptible de ser protegido por derecho de autor, en tanto los diseñadores de Java tuvieron en su momento múltiples alternativas de diseño y habrían optado en definitiva por aquella incorporada en Java.[14] La corte de apelación discrepó del parecer de la corte de primera instancia en cuanto a la aplicación de la doctrina de la fusión, por cuanto a su juicio dicha doctrina no habría restringido las decisiones creativas de los autores de Java, quienes libremente optaron por el diseño implementado. Las restricciones enfrentadas con posterioridad

---

programa computacional cuya implementación facilitaba transacciones financieras, por considerar que se trataba de la implementación de ideas abstractas no patentables. Con anterioridad, en *Ass'n for Molecular Pathology v. Myriad Genetics, Inc.*, 133 S. Ct. 2107 (2013), la Corte Suprema rechazó la patentabilidad del genoma humano que había sido declarado como patentable por el Circuito Federal.

14. Argumento que ignora que existen estándares o convenciones de programación que hacían funcionalmente más eficiente adoptar las declaraciones escogidas por los creadores de Java.

por los creadores de Android, no resultaban relevantes a ojos de la corte de apelación para decidir sobre la posibilidad de protección de las API de Java, las cuales al haber sido originales al momento de su creación eran dignas de protección bajo derecho de autor.

La corte de apelación rechazó también el razonamiento de la corte de primera instancia en cuanto a la falta de originalidad de los nombres cortos utilizados en las declaraciones de las API, considerando que tales frases cortas eran creativas al momento de su selección e incorporación en el lenguaje Java. De la misma forma, la corte de apelación consideró que la SSO de las API de Java resultaba creativa, distanciándose de la interpretación sostenida en un caso previo confirmado por la Corte Suprema, en el cual una SSO de comandos había sido precisamente identificada como un método de operación, y, por tanto, excluida de protección de derecho de autor.[15] Para la corte de apelación, la funcionalidad de una SSO no constituía un obstáculo para la protección de derecho de autor, en cuanto la SSO resultara original y creativa.

Finalmente, la corte de apelación restó toda relevancia en la decisión del alcance de la protección de derecho de autor a los argumentos de Google que apuntaban a la interoperabilidad que el uso de las API de Java hace posible. La corte reconoció que el análisis de la interoperabilidad podía ser relevante —como ha sido en casos previos—[16] para definir la aplicación del *fair use*, pero no así en relación al alcance de la protección de derecho de autor sobre las declaraciones y SSO, puesto que las opciones creativas de los diseñadores de Java no se encontraban constreñidas a la hora de diseñarlas por consideraciones de interoperabilidad. «Ya sea que la demandada luego haya perseguido hacer su programa interoperable con el programa de la demandante, ello no implica que el

15. El caso referido es *Lotus v. Borland*, 49 F.3d 807 (1995), confirmado por la Corte Suprema en 1996, en el cual Borland fue demandado por Lotus por la comercialización de una hoja de cálculo (Quattro Pro) que tenía un modo de compatibilidad en el cual su menú imitaba el producto competidor desarrollado por Lotus. Borland había replicado 469 comandos del programa Lotus y su organización, pero no había copiado nada del código del programa. La corte concluyó que en este caso los comandos y su organización constituían un método de operación no susceptibles de protección bajo derecho de autor de acuerdo a la sección 102 (b) de la Copyright Act.

16. *Sega Enterprises v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir.1992); y *Sony Computer Entertainment, Inc. v. Connectix, Corp.*, 203 F.3d 596 (9th Cir.2000).

programa creado por la demandante haya tenido cualquier limitación de diseño dictada por factores externos.»[17]

La corte de apelación dedica sus últimas palabras para aclarar que la interoperabilidad perseguida por Google al implementar las API de Java no dice relación con operar en la plataforma de Java, desarrollada y administrada por Oracle (Java Virtual Machine o JVM), sino que con capturar a los programadores ya previamente entrenados en el uso del lenguaje Java y sus API.[18] La corte de apelación considera la interoperabilidad en este sentido como una consideración competitiva ajena a la evaluación del alcance de la protección de derecho de autor,[19] cerrando los ojos a cualquier consideración sobre los efectos económicos presentes en el desarrollo de obras intelectuales, que son a la vez por naturaleza funcionales.

Como veremos en el análisis de las repercusiones de la interpretación sostenida, la noción de interoperabilidad resulta esencial de ser comprendida en una forma más dinámica, y a la luz de la realidad de cómo operan los incentivos económicos para la innovación en el desarrollo de programas de computación. Una interpretación extensiva del alcance de la protección de derecho de autor, que ignore completamente la realidad económica que exige preservar la interoperabilidad entre programas computacionales, resulta peligrosa desde la perspectiva de una política regulatoria que busque fomentar la innovación a través de limitar las barreras de entrada y reducir los costos de cambio para el desarrollo de nuevos y mejores programas computacionales que beneficien con su interoperabilidad a programadores y consumidores. La protección equi-

17. *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (2014), párrafos 1370-1371.

18. «La compatibilidad que Google perseguía incentivar no era con la plataforma de Java de Oracle o con la JVM central para esa plataforma. Por el contrario, Google quería capitalizar en el hecho de que los desarrolladores de programas se encontraban ya entrenados y tenían experiencia usando los paquetes de API de Java en discusión», *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (2014), párrafo 1371.

19. «El interés de Google era acelerar su proceso de desarrollo 'apalancándose' en Java por su base existente de desarrolladores. Aunque este objetivo competitivo podría ser relevante para la evaluación del *fair use*, concluimos que es irrelevante para el análisis de la cobertura de protección de derecho de autor de las declaraciones y la organización de los paquetes de API de Oracle», *Oracle America, Inc. v. Google Inc.*, 750 F.3d 1339 (2014), párrafos 1371-1372.

librada de los programas computacionales resulta consistente con los objetivos de propiedad intelectual de contribuir al progreso social, y no tan sólo al beneficio económico de su creadores (Samuelson, 2003).

LÍMITES DE LA PROTECCIÓN DE DERECHO DE AUTOR SOBRE LOS PROGRAMAS COMPUTACIONALES EN CUANTO OBRAS FUNCIONALES

Definir el alcance de la protección de derecho de autor aplicable a un programa computacional resulta en extremo complejo. La primera dificultad a la que nos enfrentamos es que la regulación de derecho de autor no ha sido originalmente diseñada teniendo en consideración la protección de expresiones funcionales, como lo son los programas computacionales. La segunda dificultad —que resulta corolario de la cualidad funcional de los programas computacionales— es que a riesgo de terminar protegiendo ideas o conceptos generales, debe efectuarse una disección acuciosa de aquellas expresiones creativas propiamente tales, presentes en el código del programa computacional respectivo, de aquellos elementos constitutivos que son mera aplicación de estándares de eficiencia, pero que de expresivos nada tienen. Un paso en falso en esta ponderación nos puede dejar en un escenario en el cual se entregue un monopolio de facto sobre funcionalidades, bloqueando la posibilidad de innovadores presentes y futuros de construir nuevas expresiones útiles sobre la base de las funcionalidades no protegidas. Prevenir dicho riesgo es precisamente aquello que compete a los objetivos de la regulación de propiedad intelectual (Menell, 1986).

Ahora bien, desde una perspectiva económica más amplia, una regulación de derecho de autor que bloquea la interoperabilidad no sirve a los intereses del bienestar social, pues precisamente otorga incentivos a los titulares de derechos para establecer barreras de entrada que limiten la posibilidad de innovación por terceros no relacionados, y, con ello, incrementen injustificadamente los costos de cambio de los consumidores encerrándolos involuntariamente en un determinado estándar.

La dificultad implícita en el dilema recién expuesto fue reconocida tempranamente por la jurisprudencia norteamericana a la hora de verse enfrentada a la definición del alcance de la protección de derecho de autor sobre programas computacionales. En 1995, el juez Boudin, al expresar su voto concurrente en el caso *Lotus v. Borland*, definió la di-

ficultad presentada por la protección de derecho de autor de programas computacionales distinguiéndolos de las obras de naturaleza artística:

> El programa computacional es un medio para causar que algo suceda; tiene una utilidad mecánica, un rol instrumental, completando el trabajo del mundo. Conceder protección [...] puede tener algunas de las consecuencias de la protección otorgada por las patentes limitando la posibilidad de otras personas de ejecutar una tarea en la manera más eficiente. La utilidad no limita la posibilidad de protección por derecho de autor (los diccionarios pueden ser protegidos por derecho de autor), pero altera el cálculo.[20]

La naturaleza funcional de los programas computacionales constatada por el juez Boudin, requiere que su ámbito de protección sea definido considerando la realidad económica en la cual la estandarización de la comunicación entre diferentes capas de programas computacionales resulta clave, y que es —a diferencia de una obra artística tradicional— la única razón que ha motivado su creación. Así, los elementos verdaderamente creativos de un programa computacional no pueden ser separados de los meramente funcionales sin tener en consideración las metodologías y prácticas de la ciencia de la computación.

El uso de API en programas computacionales no se relaciona con consideraciones creativas, por el contrario, se encuentra motivado por el deseo de alcanzar una estandarización que permita una amplia interoperabilidad, es decir, se trata de una decisión estrictamente funcional y no creativa. Tal como lo ilustra Curtis:

> Las interfaces de usuario son una preocupación creciente para los desarrolladores de programas computacionales porque estándares están siendo construidos en torno a ellas. Estos estándares nacen de muchas fuentes: el gobierno exige estándares para el acceso a computadores de personas con discapacidad, son motivados por el servicio a los consumidores como aquellos creados por las compañías de teléfono para equipos, estándares de facto emergen en el mercado para posibilitar que diferentes programas puedan compartir una interfaz de usuario común. Estándares de este último tipo forman la base para sistemas abiertos,

20. *Lotus Development Corp. v. Borland Intern., Inc.*, 49 F.3d 807 (1995), párrafo 819, disponible en <http://bit.ly/1Zmu9FA>.

> sistemas que comparten muchos atributos comunes, protocolos de comunicación, y formatos de datos de tal forma que los clientes pueden fácilmente interconectar diferentes computadores y programas y evitar aprender una multitud de interfaces de usuario (1989: 53).

Como lo explica en términos económicos Farrell, «la estandarización de programas computacionales facilita la formación y el uso de redes computacionales, transferencia de archivos entre usuarios y a través de aplicaciones, y genera ahorros en costos de entrenamiento» (1989: 36). La estandarización es un requerimiento de eficiencia dictado por la naturaleza funcional de los programas computacionales que requiere, para materializar dicha eficiencia, la creación de redes en las cuales tales estándares se apliquen y repliquen. En *Lotus v. Borland*, el juez Boudin fue uno de los primeros en intuir dicho fenómeno constatando la existencia de efectos de red en el diseño de programas computacionales al identificar correctamente la relevancia de la inversión hecha por los usuarios al aprender el uso de una determinada interfaz; en ese caso, el simple listado de comandos de Lotus.[21] La doctrina económica describe la existencia de efectos de red en aquellos casos en que «si la adopción por un agente de un bien: a) beneficia a otros agentes que adoptan el mismo bien (efecto total); y b) genera un incremento en el incentivo de adoptar el bien (efecto marginal)» (Farrell y Klemperer, 2006: 2007).

La relevancia que tiene la identificación de efectos de red en la determinación del ámbito de protección de derecho de autor de los programas computacionales es que tales efectos de red de alguna forma imponen el uso de elementos estandarizados para permitir que un nuevo código completamente creativo pueda comunicarse con el ecosistema de programas computacionales previamente existente a través de la utilización de un lenguaje común, y, a la vez, permita a los usuarios familiarizados con tal lenguaje capitalizar su aprendizaje en el uso del nuevo programa sin

21. «Un nuevo menú puede ser una obra creativa, pero con el tiempo su importancia puede residir más en la inversión que ha sido hecha por los usuarios aprendiendo el menú y desarrollando sus propios miniprogramas sobre el menú. Mejores disposiciones de un teclado de escritura pueden existir, pero el conocido teclado Qwerty domina el mercado porque es lo que todos han aprendido a usar», *Lotus v. Borland*, 49 F.3d 807 (1995), párrafos 819-820.

incurrir en costos de cambio que desincentiven la adopción de la nueva tecnología. Este fenómeno se ilustra en la siguiente forma:

> En el caso de las obras creativas tradicionales, tales como novelas, la protección de una creación no limita a innovadores posteriores. Si la elección de expresión del primer innovador es arbitraria, éste podría igualmente haber hecho un número ilimitado de otras elecciones, y de ello podemos ver que se sigue lógicamente que las opciones de un innovador posterior no se verán indebidamente limitadas; éste necesita sólo evitar conscientemente hacer lo mismo que el primer innovador, y esto no parece una carga particularmente problemática [...], pero este argumento falla en un mercado caracterizado por dinámicas de retornos crecientes, tales como las externalidades de red. Entonces, el solo hecho de que un innovador anterior haya usado cierta expresión arbitraria y sus clientes hayan crecido usándola, hace esa expresión arbitraria una parte importante y ya no arbitraria del diseño. Aunque, con anterioridad, el idioma inglés pudo haber sido escrito alternativamente de derecha a izquierda o de izquierda a derecha, un editor que intentara introducir esa convención ahora fracasaría de seguro (Farrell, 1989: 47-48).

El desarrollo de estándares y la masificación de su uso no es materia de selección creativa, es una elección de eficiencia. En el caso *Oracle v. Google*, el expediente contiene evidencia relevante que apoya el hecho de que Sun Microsystems desarrolló Java con el propósito directo de transformar su uso en un estándar de programación internacional. Incluso aunque Oracle no haya seguido la política construida por Sun Microsystems en esta materia,[22] los hechos —que pueden ser extraídos de los testimonios recogidos en el caso— muestran que Java se ha convertido de facto en un estándar para los programadores a nivel internacional.

22. «Existe una ironía en el reclamo de derecho de autor sobre API de Oracle contra Google. Cuando Sun era un actor principal de la industria de la computación, estaba entre los más vigorosos proponentes de la interoperabilidad en las reglas de propiedad intelectual. Sun, por ejemplo, presentó reportes de amigos de la corte en los casos Altai y Lotus tomando exactamente la posición contraria de la de Oracle en el caso Google. Sun también apoyó una exclusión de las interfaces de programas de computación de la protección de derecho de autor en Europa. Cuando una compañía compra a otra, no se encuentra obligada por las posturas legales previamente tomadas por la compañía adquirida. Pero debiera ser más juiciosa en hacer reclamos contrarios de lo que Oracle lo fue» (Samuelson, 2012: 27).

Desde la ciencia de la computación, Curtis proporciona una cándida y útil explicación de por qué el desarrollo de estándares en programas computacionales puede generar preocupaciones en cuanto a infracciones de derecho de autor:

> ¿Por qué son las interfaces de usuario similares, como es alegado en algunas demandas? [...] Primero, la organización, estructura, y secuencia de un programa y su interfaz de usuario pueden ser similares porque hay un número limitado de formas de ejecutar la tarea. [...] Mientras más programas para una aplicación particular son desarrollados, más comienza a surgir un entendimiento común acerca de la estructura más eficiente. La elección de los nombres de los comandos debería estar guiada por el vocabulario mejor entendido por los usuarios a través de un análisis de tareas o prueba de usabilidad. Por razones similares, los comandos macro y la sintaxis pueden no diferir marcadamente entre programas. Secuencias de tareas frecuentemente repetidas son candidatos lógicos para su agregación en los macro comandos (1989: 75).

El lenguaje Java desarrolló características de estándar al dar origen a externalidades de red para los programadores (efecto total), con lo cual generó el incentivo perfecto para que un innovador en servicios móviles (Google) desarrollara su sistema operativo sobre la base de un estándar ya conocido, y que permitiría a quienes adoptaran esta nueva tecnología comunicarse con los ya entrenados programadores y usuarios familiarizados con Java (efecto marginal).

La adopción de un estándar es una decisión racional cuando existen efectos de red que pueden ser capturados, como sucede en el desarrollo de programas computacionales. Como Farrell y Saloner lo explican:

> La compatibilidad es un aspecto crucial en muchas industrias, especialmente en las industrias de la información. Las telecomunicaciones habrían resultado imposibles sin compatibilidad, y así para los programas computacionales comerciales, conexiones de redes, y portabilidad se han transformado en lo más relevante comparado con la operación independiente, la industria de la computación se ha vuelto igualmente dependiente de la compatibilidad. La compatibilidad puede ser alcanzada a través de la estandarización: un acuerdo explícito o implícito para hacer ciertas cosas en una forma uniforme. La estandarización ocurre cuando los fabricantes de computadores usan las mismas inter-

> faces para la conexión de complementos, cuando las cámaras son diseñadas para usar una película común de formato de 35 mm, o cuando los diseñadores de programas computacionales adoptan una interfaz de usuario común (Farrell y Saloner, 1992: 9).

Por lo tanto, la decisión de Google de usar las API de Java, convertidas en un estándar internacional de facto, responde a una selección racional en términos de eficiencia económica para capturar los efectos de red disponibles.

El resultado más probable de conceder protección de derecho de autor a los elementos que forman parte de un estándar es la captura o *lock-in* de los usuarios que han invertido en el conocimiento y uso del estándar. Así lo explican Lemley y McGowan refiriéndose a las implicancias de la disputa entre Microsoft y Sun Microsystems en el caso de abuso de posición dominante enfrentado por Microsoft:

> Si Java cumple su promesa, podríamos encontrar que simplemente hemos cambiado Microsoft por Sun: que una vez que Java se vuelva un estándar, Sun usará su naturaleza propietaria para cerrar el estándar y actuar como un monopolista con poder permanente. Para ser justo, esto actualmente parece improbable. Sun ha reiterado en muchos diferentes foros su compromiso de servir de plataforma a la programación independiente. Sin embargo, ha insistido en retener determinados derechos de propiedad intelectual sobre Java, el vehículo propuesto para la plataforma independiente. Mientras Sun podría o no ser un monopolista más benevolente que Microsoft, la promesa real de Java está basada en que el estándar permanezca abierto, y ofreciéndonos todos los beneficios de los efectos de red y de la competencia al interior de un estándar (Lemley y McGowan, 1998a: 29-30).

El riesgo de encerrar la innovación en un estándar propietario[23] puede y debería ser mitigado por un correcto entendimiento de los límites de protección del derecho de autor sobre elementos funcionales de programas computacionales. En orden a satisfacer en mejor forma el bienestar social, la condición de estándar de facto de elementos de un programa computacional debiera ser tomada en consideración en la definición de

23. Se ha argumentado en favor de estándares propietarios y su cualidad para generar competencia; en tal sentido, véase Kaiser (2011).

la protección legal disponible cuando ello determina la forma más eficientes de alcanzar la interoperabilidad con nuevos programas.[24]

Del mismo modo, el tiempo invertido por programadores y usuarios en el aprendizaje de un estándar para poder desarrollar su uso, hace que, cualquiera haya sido la decisión creativa inicial, esa decisión no puede ser usada para bloquear la posibilidad de construir sobre el estándar aprendido. Precisamente éste ha sido el caso de todos los lenguajes conocidos por la historia de la humanidad: la selección arbitraria de un grupo de signos o letras para identificar un concepto (aquello que llamamos palabra), o la identificación de un símbolo con una operación matemática. Por supuesto que cualquiera podría reclamar que al inicio de los tiempos existió creatividad en la selección de tales expresiones, pero dicha selección tuvo como única motivación un propósito exclusivamente funcional de generar un sistema de comunicación eficiente entre los usuarios. Al igual que en los efectos de redes presentes en el uso de las API, el uso de signos estandarizados de lenguaje es esencial para que la tarea de la comunicación resulte efectiva entre los miembros de la red.

Otro tanto cabe ser afirmado respecto de las SSO, cuya estandarización previene los riesgos de *lock-in* por la vía de proveer un sistema de presentación del nuevo código que es reconocible por cualquiera familiarizado con el estándar, capturando los efectos de red provenientes de dicho conocimiento y uso previo del estándar.[25]

La búsqueda de efectos de red es aquello que motiva la implementación de estándares, pero paradójicamente son esos efectos los que pue-

24. «Todos los consumidores se beneficiarán de la masificación del acceso al estándar de red. Si la opción del consumidor está limitada por el valor inherente de una base instalada, el bienestar social se alcanzará liberando al consumidor para comprar tecnología que aunque difiera de la base instalada preserva los beneficios de la interoperación con esa base instalada. Por tanto, los regímenes legales respectivos deberían apuntar a reducir los costos de extender el estándar y la transición entre potenciales estándares» (Lemley y McGowan, 1998a: 24-25).

25. «Incluso cuando un talentoso programador ha aprendido la nueva y mejor SSO, debe escribir un código que no infrinja derechos de autor para producir el resultado deseado de la ejecución del programa, que debe adicionalmente probar y comercializar. Nadie ha demostrado que estos pasos de "ingeniería reversa" dejen a los programas computacionales más vulnerables a una copia indebida que otro producto tecnológico (dado que el código se encuentra protegido por derecho de autor)» (Karjala, 1998: 54).

den generar el *lock-in* de sus usuarios si el estándar es establecido como propietario. Luego de que un número relevante de usuarios ha invertido en el aprendizaje y uso del estándar, la posibilidad de éxito en el desarrollo de un nuevo producto estará atada en gran medida a la habilidad de incorporar el estándar en el nuevo producto. Como Katz y Shapiro lo explican:

> Mercados sistémicos presentan desafíos para la coordinación entre empresas, y a veces con los consumidores también. Una empresa que se encuentra analizando el desarrollo y lanzamiento de una nueva arquitectura de microprocesador, por ejemplo, debe saber si existirán programas computacionales disponibles para operar con el nuevo microprocesador. Del mismo modo, una empresa puede ganar poco introduciendo un nuevo formato de audio, como el cassette compacto digital, a menos que exista programación disponible para ser tocada en ese formato (1994: 94).

No es difícil entender por qué Google, en el desarrollo de un nuevo sistema operativo para un naciente mercado móvil, buscó asegurar compatibilidad a través del uso de elementos parte de un estándar ya conocido y que contaba con una masa crítica de programadores y usuarios previamente entrenados.

Desde el punto de vista de la articulación coherente de los límites de la protección de derecho de autor sobre programas computacionales, un estándar de programación no es otra cosa que un set de instrucciones o un proceso de operación que puede encontrarse incorporado en una obra original —el código del programa computacional respectivo—, pero que representa una capa utilitaria separada, y no susceptible de protección de derecho de autor, por carecer de una naturaleza expresiva y encontrarse dictado por consideraciones meramente funcionales de comunicación.

Limitar la protección de derecho de autor al código original que permite la implementación de interfaces de usuario, pero no extender dicha protección a éstas y a su organización, entendidas aquéllas como estándares de interoperabilidad —dictados por la funcionalidad y no la creatividad—, sirve al propósito de construir una protección equilibrada, que por un lado proporcione incentivo a la innovación a través de asegurar la compensación económica de quienes inviertan en la creación de

nuevo código, pero a la vez permite dejar disponibles a los innovadores futuros y usuarios los elementos que son necesarios para asegurar que el desarrollo y adopción de creaciones futuras no se vea indebidamente entrampada.[26]

Los antecedentes que forman parte de la discusión en *Oracle v. Google* muestran que las API y su organización, cuya infracción se encuentra en discusión, no son elementos susceptibles de protección bajo el derecho de autor porque forman parte de un estándar de interoperabilidad. Ellas gozan de una naturaleza meramente funcional para permitir la comunicación entre programas computacionales, permitiéndoles de esa forma capturar los efectos de redes generados en la operación dentro de un estándar. En cualquier caso, la condición de que las API de Java hayan adquirido estatus de estándar de facto no limita la posibilidad de proteger bajo derecho de autor el código de programación de Java para la implementación de tales API, o cualquier otro código de implementación desarrollado para ellas, como es precisamente aquel desarrollado por Google para Android.

En suma, el desarrollo y uso de estándares de interoperabilidad en programas de computación no se encuentra motivado por consideraciones de orden creativo, sino que exclusivamente por consideraciones de eficiencia para poder tener acceso a las ventajas competitivas de la participación de los efectos de red creados por el uso del estándar. Ello nos enfrenta a la pregunta de política pública: ¿Es la función del derecho de autor proteger como creativas meras consideraciones de eficiencia o asegurar la innovación a través de la interoperabilidad? Limitar la protección de derecho de autor de las API de Java en cuanto estándar de interoperabilidad, sirve al objetivo de una protección de derecho de autor sobre programas computacionales balanceada y que satisfaga objetivos de política pública de incentivo a la innovación, sin sacrificar la protec-

26. «Uno podría refutar la defensa tradicional de la propiedad intelectual en mercados de red, sin embargo, sí es posible demostrar que los efectos de red en sí mismos asegurarán un adecuado retorno al creador inicial incluso ausente cualquier protección de propiedad intelectual. El argumento sería que el creador de un programa parte de un mercado de red se beneficiará de la adopción adicional causada por la copia no autorizada, porque incrementa la oportunidad de que el componente se convierta en un estándar, e incremente el valor para los consumidores existentes de utilizar el producto del creador» (Lemley y McGowan, 1998b: 534).

ción del código que es aquello que resulta verdaderamente creativo y contribuye al progreso social.

¿PROTEGE A LAS API EL SISTEMA DE DERECHO DE AUTOR VIGENTE EN CHILE?

En Chile, los programas computacionales son protegidos bajo derecho de autor como obras intelectuales asimilables a las obras literarias, siempre que al igual que las obras literarias ellos cumplan con el requisito de originalidad, que se expresa por medio del código de programación (tanto fuente como objeto).[27]

En Chile, los programas computacionales no son susceptibles de ser patentados, opción que sí se encuentra disponible en otras jurisdicciones, como en Estados Unidos, país que a partir de los años sesenta comenzó a asignar patentes para la protección de programas computacionales que fuesen capaces de cumplir con las exigencias de novedad para su patentabilidad (Samuelson, 2011: 1749).

La definición dada por la ley en Chile de programa computacional pone el acento en su carácter funcional, al describirlo como conjunto de instrucciones para causar un proceso o resultado, señalando que éste es un «conjunto de instrucciones para ser usadas directa o indirectamente en un computador a fin de efectuar u obtener un determinado proceso o resultado, contenidas en un cassete, diskette, cinta magnética u otro soporte material».[28]

La mención expresa de la protección sobre el código fuente y objeto de un programa computacional, unida a la definición de programa computacional en atención a su habilidad de causar un resultado, puede dar pie a la interpretación de que bajo la ley chilena no existe una protección particular para las API, ya que la expresividad del programa se encuentra plasmada en el código fuente y objeto, que son aquellos encargados de producir el resultado o proceso que define a los programas computa-

27. La Ley 17.336 sobre Propiedad Intelectual (LPI) contiene ejemplos de obras protegidas bajo derecho de autor, mencionando entre ellas los programas computacionales. «Artículo 3. Quedan especialmente protegidos con arreglo a la presente ley: [...] 16) Los programas computacionales, cualquiera sea el modo o forma de expresión, como programa fuente o programa objeto, e incluso la documentación preparatoria, su descripción técnica y manuales de uso.»

28. Artículo 5 letra t) de la LPI.

cionales en la LPI. Como hemos visto, las API, por el contrario, son elementos de interoperabilidad entre capas de programas computacionales que acompañan el código —elemento expresivo propiamente tal—, pero que a diferencia del código son funcionales y no expresivas..

La interpretación anterior puede encontrar sustento adicional en las excepciones y limitaciones al derecho de autor reconocidas en la LPI, entre las cuales se considera la interoperabilidad y la ingeniería inversa para alcanzarla, señalando al respecto:

> Las siguientes actividades relativas a programas computacionales están permitidas, sin que se requiera autorización del autor o titular ni pago de remuneración alguna: [...] b) Las actividades de ingeniería inversa sobre una copia obtenida legalmente de un programa computacional que se realicen con el único propósito de lograr la compatibilidad operativa entre programas computacionales o para fines de investigación y desarrollo. La información así obtenida no podrá utilizarse para producir o comercializar un programa similar que atente contra la presente ley o para cualquier otro acto que infrinja los derechos de autor.[29]

La disposición autoriza la ingeniería inversa sobre copias legalmente adquiridas con la sola prohibición de reproducir y comercializar el programa respectivo, esto es, su código fuente u objeto que produce un determinado resultado, con lo cual parece excluir la protección del uso de otros elementos de interoperabilidad diferentes, como son las API.

La protección de las API bajo la normativa nacional de derecho de autor no ha sido debatida a la fecha en tribunales nacionales, en los cuales por lo general las escasas disputas referidas a la protección de programas computacionales miran a la protección del código de los mismos contra la piratería o copia íntegra, y no a la utilización de elementos particulares de interoperabilidad.

Sin embargo, algo de este debate puede verse insinuado en la discusión acontecida como parte del proceso legislativo que condujo a la incorporación de la excepción de ingeniería inversa en el año 2010 a la LPI. En la historia legislativa no existe ninguna referencia directa al tratamiento de las API, pero como parte de la discusión se recoge la opi-

29. Artículo 71 Ñ de la LPI.

nión del ejecutivo en el sentido de que la excepción de ingeniería inversa pueda servir al propósito de investigación y desarrollo en el país, sin implicar infracción de los derechos intelectuales.

La Ministra de Cultura de la época se refirió al texto propuesto (y finalmente aprobado) señalando: «con ello acota la posibilidad de investigación y desarrollo de ingeniería inversa que realizan especialmente las universidades. En la medida que se permita la investigación y el desarrollo, pero sin fines comerciales». El entonces senador Chadwick manifestó su acuerdo con la propuesta del ejecutivo en cuanto ella «acota bien en términos de que no se permita producir o comercializar el programa computacional, que es lo fundamental».[30] Como se aprecia, se busca compatibilizar la excepción con la total protección del programa computacional en cuanto a su expresión, esto es, en cuanto a su código.

Por último, la opinión pública de la época da cuenta también del propósito perseguido con la excepción, el cual justamente busca resguardar la posibilidad de estudio y uso de elementos funcionales distintos del código con el fin de generar interoperabilidad y futura innovación (Hardings, 2008).

El contexto anterior da pie a una interpretación de nuestra legislación de derecho de autor en la protección de programas computacionales que sea compatible con la total protección de sus elementos expresivos (código), preservando a su vez la posibilidad de terceros de acceder, estudiar y utilizar aquellos elementos funcionales distintos del código, entre ellas las API en comento, sin que ello constituya una infracción a los derechos de autor.

BREVE REFLEXIÓN ACERCA DE LOS RIESGOS COMPETITIVOS DE LA ADOPCIÓN DE ESTÁNDARES DE INTEROPERABILIDAD PROPIETARIOS

Oracle v. Google, un caso inicialmente atingente al alcance de la protección de derecho de autor sobre elementos de interoperabilidad como son la API de Java, es a la vez un caso que plantea preguntas relevantes acerca de la cabida que puede darse a consideraciones de competencia en la definición de la cobertura de la protección de propiedad intelectual.

30. Ambas declraciones están contenidos en la *Historia de la Ley* 20.435, Senado, 1 de septiembre, 2009, Cuenta en Sesión 45, Legislatura 357, pp. 592-593.

Sin perjuicio de que las consideraciones de interoperabilidad expresadas en el apartado anterior —destacadas como relevantes para la evaluación del alcance de la protección de derecho de autor sobre programas computacionales— incorporen de alguna forma la consideración de los beneficios competitivos de adoptar dicha interpretación, el caso proporciona adicionalmente la oportunidad para reflexionar acerca de los riesgos competitivos de la adopción de estándares de interoperabilidad propietarios, ya sea de facto, o a través de procesos formales en los cuales la falta de información acerca de la existencia de derechos intelectuales involucrados puede conducir al *lock-in* bajo el estándar seleccionado.

La adopción de Java y sus API como estándar de programación fue impulsada por Sun Microsystems, aunque la empresa inicialmente se reservó derechos propietarios sobre el código de Java, siguiendo una estrategia híbrida que fomentaba y permitía su uso libre no comercial, pero requería el pago de licencia para usos comerciales.[31] Dentro de tal régimen, resultaba ambigua la cobertura de las API dentro de los derechos intelectuales reclamados como propietarios. Con posterioridad, la empresa ofreció la licencia de código abierto del lenguaje Java, con algunas salvedades que son aquellas precisamente en disputa en *Oracle v. Google*.[32]

Como lo explica West (2003), la decisión de pasar desde estándares propietarios a abiertos puede tener múltiples motivaciones; entre ellas, la adopción masiva de facto o *tipping*[33] del estándar presiona por su aper-

31. «Originalmente en 1999 Sun ofreció el código fuente de Java bajo lo que denominó la 'Licencia Fuente de la Comunidad de Java' (Sun Community Source License o SCSL), que era un híbrido entre la licencia propietaria tradicional y una licencia de software abierto. La licencia contaba con cuatro elementos básicos: 1) derecho a modificar el código fuente; 2) libre de pago de licencia en proyectos de código abierto; 3) pago de licencias para redistribución comercial; y 4) requerimientos de prueba para mantener compatibilidad. Desde el punto de vista de Sun, la licencia provee protección para la propiedad intelectual, garantiza innovación estructurada en una sola organización responsable, [y provee] claro control sobre compatibilidad» (West, 2003: 1276).

32. En 2006 Sun MicroSystems ofreció bajo una licencia de software abierto (GPLv2) su lenguaje de programación Java, a través de OpenJDK (Open Java Development Kit) (Martens, 2006). Sin perjuicio de lo anterior, Sun mantuvo como propietario el código que permite la certificación de un software como parte de Java Virtual Machine (JVM).

33. «En mercados con efectos de red, existe una tendencia natural hacia la estanda-

tura, y hace imposible establecer o mantener un estándar propietario. Conforme explica el mismo autor, a través del uso de estrategias híbridas de apertura los titulares de derechos intelectuales intentan mantener las ventajas competitivas de un estándar propietario, sin enfrentar el rechazo del mercado que puja por la apertura. La adopción de este tipo de estrategias híbridas puede tomar la forma de liberar el control de ciertas capas, que se consideran *commodities*, mientras se retiene el control de otras capas que pueden proveer más diferenciación; o bien, proporcionar un acceso abierto parcial a los clientes, pero restringir el acceso a los competidores. Tales estrategias dan cuenta de la actitud ofensiva de la compañía que promueve el estándar por impulsar su masificación, a través de hacerlo más atractivo para competidores y usuarios estratégicos; en el caso de Java, los programadores. La ventaja que dicha estrategia puede proveer es incrementar el número de productos interoperables en el estándar, que como hemos visto resulta relevante en mercados con efectos de redes como son los programas computacionales (West, 2003: 1279-1280).

El proceder de Sun, y luego de Oracle en relación a la apertura de Java y sus API como estándar de programación, calza totalmente con la descripción anterior de incentivos económicos, explicando la racionalidad de la conducta en el impulso por transformar Java en un estándar de programación. El problema que presenta la disputa entre Oracle y Google es el intento por vía de protección de derecho de autor de evadir las consecuencias de la estrategia híbrida implementada, incluyendo las API adoptadas como estándar de facto en las capas propietarias reservadas sobre el código de Java.

El problema del *lock-in* creado por estándares de interoperabilidad propietarios no es ajeno al análisis de la doctrina y autoridades antimonopolio a nivel internacional.[34] Como ha sido analizado en el apartado

---

rización de facto, lo que significa que todos usan el mismo estándar. Por causa de los fuertes elementos de retroalimentación, los mercados sistémicos están particularmente inclinados al *tipping*, que es la tendencia de un sistema de alejarse de sus rivales en popularidad una vez que ha alcanzado masa crítica inicial» (Katz y Shapiro, 1994: 105-106).

34. En general, la interacción entre la normativa de propiedad intelectual y el derecho de la competencia es un tópico que concita relevante interés en la doctrina internacional. Para un panorama general de las más relevantes de tales interacciones, véase Hovenkamp (2013).

anterior, en mercados con fuerte presencia de efectos de red el camino hacia la estandarización está marcado por la búsqueda de la captura de eficiencias.[35] El problema es que la captura efectiva de tales eficiencias, en una forma alineada con el beneficio social, está determinada en gran medida por dos variables: el proceso de definición del estándar, y la existencia o no de elementos propietarios en el estándar escogido (generalmente alguna forma de propiedad intelectual: patentes o derechos de autor).

En términos de proceso, el mercado será el campo de prueba natural para la determinación de la eficiencia de un estándar sobre otro: la búsqueda de los efectos de redes privilegiará la adopción de estándares que tiendan a la interoperabilidad proporcionado a los desarrolladores de nuevos productos y a los consumidores la posibilidad de escoger libremente entre diferentes plataformas de operación sin incurrir en costos de cambio.

Sin embargo, en ocasiones la existencia de elementos propietarios en los estándares en competencia hace más cauteloso el alcance de consenso en el mercado, en el que cada cual intenta esquivar el riesgo de *lock-in*. En tales casos, el consenso puede ser alcanzado a través de la gestión de alguna organización privada de estandarización, la cual invita a todos los interesados en el desarrollo del estándar a participar de su proceso de definición, proporcionando información técnica de las diferentes alternativas para construir el estándar, así como entregando información acerca de la existencia o no de elementos propietarios en el estándar propuesto, y de haberlas, las formas en las cuales a través de licencias justas, razonables y no discriminatorias tales elementos propietarios serán

35. «Estas diversas estrategias [hablando del cambio desde plataformas propietarias hacia abiertas] reflejan la tensión esencial de la creación de estándares de facto: que está entre la apropiabilidad y la adopción. Para recuperar los costos del desarrollo de la plataforma, su patrocinador debe ser capaz de apropiarse para sí mismo una parte de los beneficios económicos de la plataforma. Pero para obtener retornos, el patrocinador debe lograr la adopción de la plataforma, lo cual requiere compartir los retornos económicos con compradores y otros miembros de la cadena de valor. Las estrategias propietaria y abierta corresponden a los dos extremos de este *trade-off*. En el diseño de la estrategia para una plataforma en el siglo XXI, los líderes de la venta de programas computacionales enfrentan el dilema de determinar cuánta apertura es suficiente para atraer a los compradores y a la vez asegurarse retornos adecuados» (West, 2003: 1259).

licenciados entre quienes deseen operar en el estándar. El procedimiento aquí descrito de las organizaciones de estandarización —que ha sido refrendado por las autoridades de competencia en los Estados Unidos[36] y valorado positivamente por la doctrina—[37] permite prevenir que la adopción de un estándar propietario genere el *lock-in* de desarrolladores de la tecnología y usuarios.

El problema presentado en el caso del lenguaje de programación Java es que su calidad de estándar de facto fue impulsada por la creencia —alimentada por las acciones de Sun Microsystems— de programadores y usuarios acerca de la apertura del lenguaje de programación y la disponibilidad de uso de sus API sin restricciones propietarias. El control propietario sobre las API de Java, reclamado posteriormente por Oracle, le otorga la habilidad y el incentivo para bloquear la innovación de terceros en el desarrollo de programas computacionales. Lo anterior se traduce en dos riesgos competitivos claros de daño al bienestar social: producción de productos ineficientes en el mercado,[38] o bien permite a Oracle ejercer su posición de monopolista sobre el estándar para apro-

36. Por ejemplo, en *Allied Tube*, 486 U.S. at 500-01, 108 S.Ct. 1931, *Dell Computer Corp.*,121 F.T.C. 616 (1996), *Rambus Inc. v. FTC*, 522 F.3d 456 (D.C. Cir. 2008) y *Broadcom Corp. v. Qualcomm Inc.*, 501 F. 3d 297 (2007).

37. «Un confuso orden privado dictado por SSO [organizaciones de estandarización] puede que sea o no mejor para la innovación que un sistema óptimamente diseñado de propiedad intelectual. Pero es casi seguro que es mejor que el problemático sistema de propiedad intelectual que hoy tenemos. A través de reducir algunas de las amenazas a la innovación de patentes excesivamente amplias y que se sobreponen, las reglas de propiedad intelectual de las SSO ayuda al sistema de propiedad intelectual a lograr aquello para lo que originalmente fue diseñado: promover la innovación» (Lemley, 2002: 1972).

38. En tal sentido para el caso de Java: «Que el valor social de internet descansa en su habilidad de facilitar la interoperabilidad, y que internet se encuentra sujeta a efectos de red tanto en su infraestructura como en los programas computacionales usados para darle vida, implica una fuerte tendencia a la estandarización. Esa tendencia por su parte aboga por un acceso abierto al estándar o, dicho de otra forma, un «estándar» de interoperabilidad. Tal resultado puede ser alcanzado a través de la alteración de los derechos de propiedad intelectual concernidos, a través de la comercialización de productos propietarios que faciliten la interoperabilidad, o quizás ambos. Idealmente, sistemas abiertos como Java triunfarán en la competencia en el mercado, y la ley no necesita hacer nada más que fiscalizar los estándares de competencia para asegurarse que el estándar abierto no es en alguna forma subvertido» (Lemley y McGowan, 1998: 51).

piarse de una parte de las ganancias generadas por las innovaciones que se desarrollen conforme al estándar a través de la exigencia de licencias.

La última de tales motivaciones es aquella que parece verse reflejada en la postura de Oracle en cuanto a la protección bajo derecho de autor de las API de Java. Es un argumento encaminado a asegurar que por vía de la exigencia de licencias para el uso de elementos de interoperabilidad, Oracle pueda capturar parte de la rentabilidad que ha significado el desarrollo de Android como sistema operativo móvil por Google, que ha desarrollado con éxito un mercado para aplicaciones móviles para programadores entrenados en el lenguaje Java.

Desde la perspectiva de la propiedad intelectual, lo anterior parece razonable, ya que precisamente el objetivo de ésta es conceder al titular de los derechos intelectuales la autorización legal para recoger sus ganancias monopólicas sobre la propiedad inmaterial protegida. El problema radica en que cuando aquello protegido son elementos funcionales que integran un estándar que define la interoperabilidad, sin que la característica propietaria de tales elementos haya sido conocida, discutida y voluntariamente aceptada por quienes han tomado parte en la innovación utilizando el estándar atraídos precisamente por las cualidades de interoperabilidad ofrecidas.

Este problema generado por erróneos entendimientos acerca de la naturaleza propietaria de un estándar de facto no es nuevo en el mercado de los programas computacionales. Un incidente notable a este respecto fue aquél que involucró en los años noventa al algoritmo responsable de GIF, el primer formato popular de compresión de imágenes. A pesar de encontrarse protegido por una patente, el algoritmo fue revelado por uno de sus inventores en una revista de computación, y fue en general ampliamente promocionado por la empresa titular de la patente como libremente disponible.

Luego de convertirse en el estándar de facto para la distribución de imágenes a través de la naciente internet, los titulares de derechos sobre la patente respectiva comenzaron a exigir el pago de licencias por su uso a compañías desarrolladoras de software. La industria intentó escapar del *lock-in* a través del impulso de un nuevo estándar, sin embargo, la masiva aceptación de GIF como estándar transformó en infructíferos tales esfuerzos. Aunque no existió una clara intención de fraude del titular de derechos intelectuales sobre el estándar, el efecto de la ambigüedad de

la conducta respecto del ejercicio de tales derechos condujo a la industria (programadores y también usuarios) a adoptar el estándar sobre un erróneo entendimiento acerca de su apertura (Duane, Matthew, 2006: 118-122).

Desde la perspectiva de la normativa antimonopolio, han existido casos recientes a nivel internacional en los cuales la adopción de estándares ha sido denunciada por la existencia de conductas inductivas de error por parte los titulares de derechos intelectuales. Si bien la particularidad de tales casos es que ellos se desarrollaron en el marco de procesos impulsados por entidades de estandarización, el razonamiento acerca de los riegos competitivos del desarrollo de un estándar —que envuelve inadvertidamente elementos propietarios— es teóricamente posible de ser aplicado en caso de estándares de facto cuya promesa de apertura es traicionada con posterioridad.

En el caso *FTC v. Dell*, Dell participó en el proceso de estandarización de elementos de comunicación para la conexión de dispositivos complementarios (módems, terminales de video, impresoras, etcétera) desarrollado por la Video Electronics Standards Association (VESA), sin entregar información acerca de la titularidad de patentes sobre parte de los elementos de la tecnología propuesta como estándar. Con posterioridad, Dell comenzó a exigir el pago de regalías de los fabricantes que adoptaron el estándar creado. El caso fue resuelto a través de un acuerdo con la Federal Trade Commission (FTC) en el cual Dell aceptó licenciar la tecnología en forma gratuita.[39] En una línea similar, en 2007 Qualcomm fue imputada por las infracción de la normativa antimonopolio por su falla en la entrega al Instituto Europeo de Estándares de Telecomunicaciones, y su equivalente en Estados Unidos, de la información acerca de su titularidad de propiedad intelectual sobre elementos de los chips para teléfonos móviles incluidos en el estándar, seguido por su consecuente intento de recolectar el pago de licencias.[40]

Un resultado completamente distinto a los anteriores fue alcanzado en el caso *Rambus v. FTC*, referido a la falta de información por parte de Rambus acerca de sus derechos de propiedad intelectual sobre tecnología de tarjetas de memoria (Dynamic Random Access Memory chips

39. *Dell Computer Corp.*,121 F.T.C. 616 (1996).

40. *Broadcom Corp. v. Qualcomm Inc.*, 501 F. 3d 297 (2007).

o DRAM) durante el proceso de definición de estándares ante la organización de estandarización de semiconductores, el Joint Electron Device Engineering Council o JEDEC. En este caso, la corte terminó por absolver a la empresa porque no se acreditó que la falta de información acerca de la titularidad de derechos intelectuales habría sido suficiente para que la organización de estandarización tomara una decisión diferente. Adicionalmente, la corte estimó que la falta de oportunidad para negociar los términos razonables y equitativos de una licencia no resultaba un daño competitivo que proporcionara sustento a una acción en sede antimonopolio.[41] El mismo caso fue terminado en Europa a través de un acuerdo en el cual la compañía aceptó reducir los pagos de las licencias por la tecnología.[42]

El estándar de causalidad exigido en Rambus por la corte norteamericana difiere esencialmente de aquél sostenido en el caso Microsoft. En este último, la conducta acreditada de Microsoft de bloquear el acceso y la interoperabilidad a las API desarrolladas para su sistema operativo precisamente a productos de Sun Microsystems (en lenguaje Java), fue considerada por la corte como una con suficiente probabilidad de daño al mercado a través de la generación de barreras de entrada a la innovación como para dar por acreditada la conducta anticompetitiva.[43]

En Europa, el caso Microsoft parece haber influenciado en forma relevante el enfoque de persecución de ilícitos anticompetitivos en mercados caracterizados por efectos de red y búsqueda de estandarización. Es así que el documento «Orientaciones obre las prioridades de control de la Comisión en su aplicación del artículo 82 [hoy 102] del Tratado CE a la conducta excluyente abusiva de las empresas dominantes»,[44] ca-

41. *Rambus Inc. v. FTC*, 522 F.3d 456 (D.C. Cir. 2008).

42. En diciembre de 2009, la Comisión Europea aprobó el compromiso ofrecido por Rambus de poner un límite a los *royalties* cobrados por las patentes para chips de memoria de computadores. Inicialmente la Comisión acusó a Rambus de infringir las normas de competencia al cobrar *royalties* por tales patentes que no habían sido informadas en el proceso de definición del estándar para la confección de tales chips. Véase Schellingerhout y Cavicchi (2010).

43. Ver *United States v. Microsoft*, 253 F.3d 34 (D.C. Cir. 2001).

44. En 2009, la Comisión Europea publicó esta guía acerca de sus prioridades de persecución en casos de abuso de posición dominante. Este documento tiene una naturaleza *sui generis*, ya que no pretende reescribir la normativa aplicable, cuya definición

racteriza la negativa de licenciar información de interoperabilidad como un caso específico de negativa a contratar,[45] señalando a este respecto que la negativa puede no ser absoluta, sino también constructiva en el sentido de hacer poco razonable las condiciones de acceso o cambiar irrazonablemente tales condiciones.[46]

A juicio de alguna doctrina, la comunicación en comento sugiere que los deberes de compañías dominantes en este tipo de mercados podría ir tan lejos como encontrarse obligadas a proveer información de interoperabilidad a sus competidores y apoyar el desarrollo de estándares abiertos (Czapracka, 2007: 103). Este nivel más exigente de escrutinio de libre competencia en la industria se justificaría en el hecho de que este sector se caracteriza por extensivas relaciones verticales y efectos de red. Estos últimos pueden constituir barreras significativas y conducir a un *lock-in* colectivo en una tecnología establecida. Así las cosas, parecería clave el rol de un monopolista en este sector en control de un estándar de facto, cuya ventaja podría venir dada por su posición de innovador inicial en el mercado (Czapracka, 2007: 104).

En suma, aun cuando a la fecha no existen precedentes de acciones en sede antimonopolio por conductas ambiguas en cuanto a elementos propietarios en estándares de facto, tal como lo ilustran los casos acontecidos en el marco de organizaciones de estandarización, la posibilidad de considerar la conducta de falta de transparencia de un titular de derechos de propiedad intelectual sobre elementos constitutivos de

---

es competencia exclusiva de la Corte de Justicia Europea, pero explica las circunstancias bajo las cuales la conducta de una empresa con posición dominante es probable que sea objeto de persecución por parte de la Comisión Europea.

45. «El concepto de denegación de suministro abarca una amplia gama de prácticas, tales como la denegación de suministro de productos a clientes nuevos o existentes (3), la denegación de concesión de licencias sobre derechos de propiedad intelectual (4), lo que incluye, cuando la licencia sea necesaria, la información sobre interfaces (5), o la denegación de concesión de acceso a una instalación esencial o a una red (6)» (Comisión Europea, 2009: 18).

46. «Del mismo modo, no es necesario que exista una verdadera negativa por parte de una empresa dominante: basta con una «negativa constructiva». La negativa constructiva podría, por ejemplo, consistir en demorar indebidamente o en degradar de cualquier otro modo el suministro del producto o en imponer condiciones ilógicas a cambio del suministro» (Comisión Europea, 2009: 79).

un estándar en programas computacionales como un ilícito de competencia dependería de: 1) la existencia de una conducta inductiva a error del titular de derechos en cuanto a sus pretensiones propietarias sobre el estándar; 2) que dicha conducta le otorgue una posición dominante sobre cualquiera que opere en el estándar una vez generado el efecto de *lock-in*; 3) la posibilidad de acreditar un riesgo concreto de daño en el mercado, ya sea por la habilidad del titular de tales derechos de extraer de desarrolladores (competidores) y usuarios rentas supracompetitivas, o de generar barreras de entrada para la innovación a partir del control sobre el estándar; y 4) la posibilidad de acreditar la existencia de una relación causal entre la conducta ambigua o inductiva a error y el daño al mercado.

CONCLUSIONES

Oracle v. Google nos enfrenta a la dificultad de definir los límites de la protección de derecho de autor sobre obras funcionales, como son los programas computacionales. Las normas tradicionales de derecho de autor no están bien equipadas para resolver los límites de la protección que corresponde a los elementos no expresivos, sino meramente funcionales de los programas computacionales.

La necesidad de preservar la interoperabilidad entre programas computacionales responde a la realidad económica de éstos, caracterizada por la presencia de economías de red. El reconocimiento de tal realidad económica es esencial para fijar el límite de la protección de derecho de autor en una forma que preserve la interoperabilidad que ella exige. Ello porque el fin social último del derecho de autor es promover la innovación y el acceso de la sociedad a los bienes intelectuales fruto de dicha innovación.

Estándares de interoperabilidad establecidos de facto, que contienen elementos funcionales protegidos por derecho de autor (propietarios), sin que ello haya sido claramente conocido y aceptado al momento de su adopción, amenazan con encerrar a los desarrolladores y usuarios. Frente a dicho resultado, eventualmente la conducta podría ser cuestionada en sede antimonopolios cuando ha sido el titular de los derechos de autor sobre los elementos funcionales quien ha creado el *lock-in* en el estándar con su conducta inductiva a error, que le otorga una posición

dominante en el mercado que lo provee de la habilidad y el incentivo para abusar de competidores y usuarios por la vía de exigencia del pago de licencias.

El marco teórico aquí expuesto pretende abrir el cuestionamiento acerca de la necesidad de incorporar consideraciones económicas en la definición del ámbito de protección de los derechos de autor sobre obras funcionales. Sin perjuicio de la eventual aplicación de los remedios *ex-post* a conductas que resulten anticompetitivas que aquí se proponen, parece más eficiente desde un punto de vista de política pública el escoger un nivel de protección que resulte consistente con la realidad económica de los programas computacionales en cuanto obras funcionales, permitiéndoles así cumplir con la promesa de progreso social que toda protección de propiedad intelectual supone promover.

REFERENCIAS

Comisión Europea (2009). «Comunicación de la Comisión: Orientaciones sobre las prioridades de control de la Comisión en su aplicación del artículo 82 del Tratado CE a la conducta excluyente abusiva de las empresas dominantes». *Diario Oficial Unión Europea*, 24 de febrero, pp. 7-20. Disponible en <http://bit.ly/1ZmuyYw>.

Curtis, Bill (1989). «Engineering computer "look and feel": User interface technology and human factors engineering». *Jurimetrics Journal*, 30 (1): 51-78. Disponible en <http://bit.ly/1SPYWLe>.

Czapracka, Katarzyna (2007). «Where antitrust ends and IP begins. On the roots of the transatlantic clashes». *Yale Journal of Law & Technology*, 9 (1): 44-108. Disponible en <http://bit.ly/2023csT>.

Duane, Matthew (2006) «For the people and by the people: A new proposal for defining industry standards in computer software». *Wake Forest Intellectual Property Law Journal*, 7 (1): 97-143. Disponible en <http://bit.ly/1RnzRHZ>.

Farrell, Joseph (1989). «Standardization and intellectual property». *Jurimetrics Journal*, 30 (1): 35-50. Disponible en <http://bit.ly/1RnzXzd>.

Farrell, Joseph y Paul Klemperer (2006). «Coordination and lock-in: Competition with switching costs and network effects». En M.

Armstrong y R. Porter (eds.), *Handbook of industrial organization*. Elsevier: Amsterdam. Disponible en <http://bit.ly/231liO4>.

Farrel, Joseph y Saloner Garth (1992). «Converters, compatibility, and the control of interfaces». *The Journal of Industrial Economics*, 40 (1): 9-35. Disponible en <http://bit.ly/1P2y5vF>.

Gordon, Wendy y Robert Bone (2000). «Copyright». En B. Bouckaert y G. DeGeest (eds.), *Edward Elgar, Encyclopedia of Law & Economics*, 2: 189-223. Disponible en <http://encyclo.findlaw.com/1610book.pdf>.

Hardings, Jens (2008). «¿Por qué la ACTI pretende castrar la competitividad de las empresas que desarrolla(ría)n software en Chile?». *Manzana Mecánica*. Disponible en <http://bit.ly/1l9KX58>.

Hovenkamp, Herbert (2013). «Innovation and competition policy». En *Innovation, IP rights, and anticompetitive exclusion*. University of Iowa Legal Studies Research Paper. Disponible en <http://ssrn.com/abstract=1946379>.

Kaiser, Hanno (2011). «Are 'closed systems' an antitrust problem?». *Competition Policy International*, 7 (1): 91-113. Disponible en <http://ssrn.com/abstract=1844944>.

Katz, Michael y Carl Shapiro (1994). «Systems competition and network effects». *Journal of Economic Perspectives*, 8 (2): 93-115. Disponible en <http://socrates.berkeley.edu/~scotch/katz_shapiro.pdf>.

Lemley, Mark A. y David McGowan (1998a). «Could Java change everything? The competitive propriety of a proprietary standard». *Antitrust Bulletin*, 43: 715-773. Disponible en <http://ssrn.com/abstract=57515>.

—. (1998b). «Legal implications of network economic effects». *California Law Review*, 86 (3): 479-611. Disponible en <http://bit.ly/1OTisEI>.

Lemley, Mark. (2002). «Intellectual property rights and standard-setting organizations». *California Law Review*, 90: 1889-1890. Disponible en <http://bit.ly/1nkWUXo>.

Martens, China (2006). «It's official: Sun open sources Java». *Java World*. Disponible en <http://bit.ly/1Q7Uxm8>.

Menell, Peter (1986). «Tailoring legal protection for computer software». *Stanford Law Review*, 39: 1329-1372. Disponible en <http://bit.ly/1N9jt67>.

Samuelson, Pamela (2003). «Should economics play a role in copyright law and policy? ». *University of Ottawa Law & Technology Journal*, 1: 1-21. Disponible en <http://bit.ly/1Q7UQx8>.

—. (2011). «The uneasy case for software copyrights revisited». *George Washington Law Review*, 79 (6): 1746-1782. Disponible en <http://bit.ly/1TXButP>.

—. (2012). «Oracle v. Google: Are APIs copyrightable?». *Communications of the ACM*, 55 (11): 25-27. Disponible en <http://scholarship.law.berkeley.edu/facpubs/2354>.

Schellingerhout, Ruben y Piero Cavicchi (2010). «Patent ambush in standard-setting: the Commission accepts commitments from Rambus to lower memory chip royalty rates». *Competition Policy Newsletter* (European Commission), (1): 32-36. Disponible en <http://bit.ly/1RPkG9f>.

Sliwa, Carol (1999). «Sun abandons Java standards effort». *Computerworld*. Disponible en <http://bit.ly/1TXBOc5>.

Van Houweling, Molly (2005). «Distributive values in copyright». *Texas Law Review*, 83 (6): 1535-1579. Disponible en <http://bit.ly/1OsTHM5>.

West, Joel (2003). «How open is open enough? Melding proprietary and open source platform strategies». *Research Policy*, 32 (7): 1259-1285. Disponible en <http://bit.ly/2025bol>.

SOBRE LA AUTORA

María Paz Canales Loebel es abogada. Licenciada en Ciencias Jurídicas y Sociales por la Universidad de Chile. LL.M in Law and Technology, University of California, Berkeley, School of Law. Fellow 2015-16 en el Berkman Center for Internet & Society, Harvard University, Estados Unidos. Su correo electrónico es mpcanales@gmail.com y su dirección postal es 23 Everett Street, Cambridge, Massachusetts, Estados Unidos.

Artículo recibido el 2 de octubre de 2015 y aprobado el 25 de noviembre de 2015.