EL *SOFTWARE* LIBRE Y SUS IMPLICACIONES JURÍDICAS

Gladys Stella Rodríguez*

* Abogada, Magíster en Planificación y Gerencia de Ciencia y Tecnología. Doctora en Derecho, Profesora asociada e investigadora adscrita al Instituto de Filosofía del Derecho Dr. J.M. Delgado Ocando de la Facultad de Ciencias Jurídicas y Políticas de la Universidad del Zulia (Venezuela).
Correspondencia: Av. 4 Bella Vista con intersección Avenida Universidad, Edificio Residencias Mercedes Plaza, Piso No. 1D, Maracaibo, Estado Zulia (Venezuela).
Origen de subvenciones y apoyos recibidos: Versión de un capítulo del trabajo de ascenso a Profesora Titular, parte de un proyecto no financiado con el apoyo del Instituto de Filosofía del Derecho de la Universidad del Zulia (Venezuela).

REVISTA DE DERECHO
Nº 30, Barranquilla, 2008
ISSN: 0121-8697

Resumen

En esta investigación se describirán las características del software libre *y sus implicaciones legales. En este escenario, no se trata de desconocer el esfuerzo intelectual de quienes producen tecnología* (hardware, software, *conocimientos, entre otros). Sólo que esta práctica se ha desnaturalizado creando riesgos de mantenerse el conocimiento y su acceso a un grupo selecto hegemónico. En consecuencia, hoy existe la posibilidad de que a través de licencias públicas generales se consagren derechos a los autores, en un marco de mayor flexibilidad* (copyleft). *La metodología es descriptiva - explicativa, con una revisión bibliográfica y hemerográficas sobre el tema. Asimismo, se consultó a Fundación para la Ciencia y la Tecnología (FUNDACITE ZULIA), a través de la Misión Ciencia. Se concluye que el* software libre *es una oportunidad para democratizar el acceso a la información en países en desarrollo, y es un reto para el desarrollador de* softwares.
Palabras clave: *Software* libre, implicaciones legales, licencias públicas, copyleft.

Abstract

With this research, will describe the characteristics of free software and its legal implications. In this scenario, it is not unaware of the intellectual effort of those producing technology (hardware, software, expertise, among others). Except that is practiced has been distorted by creating risks to keep its knowledge and access to a select group of hegemony. As a result, today there is a possibility that through general public license rights to be enshrined authors, in a framework of greater flexibility (copyleft). The methodology is descriptive - explanatory, with a review Bibliographic and library records on the subject. It was also consulted Foundation for Science and Technology (FUNDACITE ZULIA), through the Mission Science. Concluded that free software is an opportunity to democratize access to information in developing countries, remains a challenge for the software developer.
Key words: Free software legal implications, public licenses, copyleft.

Fecha de recepción: 4 de marzo de 2008
Fecha de aceptación: 23 de octubre de 2008

1. GENERALIDADES

Hoy la realidad nos acerca a cambios de paradigmas en lo que tiene que ver con el propio derecho de propiedad intelectual y, particularmente, con el derecho de autor frente al desarrollo de las Tecnologías de Información y Comunicación (en adelante TICs), pues la idea de protección de la creación intelectual se enfrenta con los reclamos de miles de personas que proclaman un uso democrático de la red y de todos los bienes que existen en ella.

En consecuencia, el mismo proceso de desarrollo tecnológico ha conllevado a presentar alternativas de uso sin exclusión de algunas obras en la red, y a los efectos del presente trabajo nos referiremos al *software libre*. Frente a esta nueva modalidad de obra, verificaremos su definición, su significado, sus desafíos y oportunidades.

Los términos *software libre* (*free software*) y *software* de fuentes abiertas (*open source software*) son utilizados por los partidarios de este movimiento, en contraposición a la denominación *software* propietario o de dominio privado, asignado a los programas de uso restringido o autorizado, y al cual haremos algunas alusiones en este trabajo.

De manera general estos términos se relacionan con el modelo de desarrollo y distribución del *software* creado cooperativamente, cuyos códigos del sistema, o cada uno de los programas, en lugar de constituir secretos celosamente guardados son puestos a disposición de los usuarios.

Específicamente, de acuerdo con Argudo (2004), la expresión *software libre* se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*. De modo más preciso, se refiere a cuatro libertades de los usuarios del *software*:

- ejecutar el programa para cualquier propósito;
- estudiar el funcionamiento del programa para adaptarlo a cualquier necesidad;
- redistribuir copias; y
- mejorar el programa y poner las mejoras a disposición del público.

La libertad para usar un programa significa la posibilidad que tienen todas las personas u organizaciones de utilizarlo en cualquier tipo de sistema informático y realizar la clase de trabajo que fuere, sin la obligación de comunicar dicho uso y disposición al desarrollador o algún titular específico.

Se atribuye la libertad de hacer modificaciones y utilizarlas de manera privada en el trabajo u ocio, sin la obligación de anunciar los cambios ni el requerimiento de una autorización en particular.

El usuario tiene la libertad de distribuir copias, sea con o sin modificaciones. Esto incluye tanto las formas binarias o ejecutables del programa, como su código fuente, sean versiones modificadas o sin modificar (distribuir programas de modo ejecutable es necesario para que los sistemas operativos libres sean fáciles de instalar).

Para que el ejercicio de las libertades de realizar modificaciones y publicar versiones mejoradas del programa tengan sentido, se tendrá acceso al código fuente del programa. Por lo tanto, el acceso al código fuente es una condición necesaria para que se hable de *software libre*.

Bajo este esquema están desarrollados tanto *Linux* como la mayor parte de programas que con él podemos correr, así como otros sistemas operativos. Las aplicaciones más famosas del *software libre*, entre otras, son:

- el sistema operativo *Linux*
- el servidor de web Apache
- el manejador de bases de datos objeto-relacional *PostgreSQL*
- el navegador *Mozilla*
- la *suite* de aplicaciones de escritorio *OpenOffice*
– el servidor de correo *Sendmai*

HISTORIA DEL *SOFTWARE* LIBRE

El *software* como tal nació libre y se mantuvo libre hasta que surgió el interés de tratarlo como un producto que se puede comercializar.

Realmente el *software* libre es más antiguo que el denominado propietario, ya que en los años sesenta y setenta no se consideraba un producto sino un añadido que los vendedores de grandes computadoras aportaban a sus clientes para que éstas pudieran usarlas. En dicha cultura, era común que los programadores y desarrolladores de *software* compartieran libremente sus programas unos con otros. Este comportamiento era particularmente habitual en algunos de los mayores grupos de usuarios de la época, como DECUS (grupo de usuarios de computadoras DEC). A finales de los setenta las compañías iniciaron el hábito de imponer restricciones a los usuarios, con el uso de acuerdos de licencia (Wikipedia, 2008, consultado: 5-01-08).

Solo hasta mediados de los ochenta, Richard Stallman formalizó las ideas básicas del movimiento del *software* libre que está revolucionando la industria del *software*. El *software* libre, tal y como lo conocemos hoy, dio sus primeros pasos con un manifiesto a favor de la libertad de expresión y un proyecto conocido hoy mundialmente, el proyecto GNU. (www.cnti.gob.ve, Consultado: 07-01-08). Pero la fecha de nacimiento de la comunidad F/OSS (*Free/Open Source Software*)[1]. Y sus hechos fundamentales pueden ser determinados de una manera muy breve como la siguiente. En efecto, el nacimiento de lo que podemos denominar *software libre* (*free software*), la parte política más radical de la comunidad se remonta a mediados de la década de los ochenta, cuando Richard M. Stallman concibe la idea de lo que se conoce como Proyecto GNU, funda la *Free Software Foundation (FSF)* y redacta el acta fundacional de la comunidad: El Manifiesto GNU.

Por su parte, el surgimiento de la otra "parte política" del movimiento, conocida como *software* de fuente abierta (*open source software)*, se remonta a una época mucho más cercana, más precisamente a la segunda parte de la década de los noventa, cuando un grupo de partidarios

[1] F/OSS: Una de las dificultades al hablar de *software* libre atañe a la terminología que se debe emplear para referirse a la comunidad como un todo y a cada una de sus partes; se adopta por ello la expresión comunidad *Free/Open Source Software (F/OSS) para referirnos tanto a free software u open source software.*

del movimiento se reúnen con la finalidad de comenzar una campaña de promoción del *software* libre para llevarlo al ámbito empresarial, y propone como estrategia fundamental el reemplazo del término *free software* por *open sorce software.*

Por su parte, la historia de la FSF y del Proyecto GNU ha sido relatada por su fundador, R.M. Stallman, y la podemos resumir así:

a) **La *Free Software Foundation* y el Proyecto GNU**

En 1971, Stallman había ingresado al *Laboratorio de Inteligencia Artificial del Massachussets Institute of Technology* (MIT), en el que se acostumbraba a compartir el código fuente[2] del *software* que se utilizaba. El ingreso de Stallman al MIT coincide en el tiempo con la obligada decisión de IBM de separar el *software* del *hardware* y, por consiguiente, con los inicios del *software* propietario.

En 1984 Stallman, en completa disconformidad con la idea de que el *software* tuviera propietarios, idea que en ese momento ya había sido ampliamente aceptada en el mundo del *software*, renuncia a su trabajo en el MIT con la intención de crear y desarrollar un sistema operativo completo, estilo UNIX[3], que pudiera ser usado, estudiado, copiado, modificado o redistribuido por cualquiera, libremente, y formar una comunidad de desarrolladores en torno a éste. Stallman quería evitar la posibilidad de que el MIT, alegando que él formaba parte del *staff*, reclamara derechos de propiedad sobre su futura obra y la transformara en un paquete propietario.

---

[2] CÓDIGO FUENTE: (*Source code, code base*). Texto escrito en un lenguaje de programación específico y que puede ser leído por un programador. Debe traducirse a lenguaje máquina para que pueda ser ejecutado por la computadora o a *bytecode* para que pueda ser ejecutado por un intérprete.

[3] UNIX: Es un sistema operativo desarrollado desde hace 30 años, y conserva el mismo diseño y forma de uso, añadiendo diversas mejoras a lo largo de los años, pero manteniéndose sobre la línea de seguridad y alto rendimiento sobre la cual fue diseñado. TCP / IP: El protocolo básico de Internet; fue construido alrededor de UNIX, por lo tanto la integración de los servicios de Internet en un ambiente UNIX es perfecta.

Naturalmente, dado que la intención de Stallman era desarrollar un sistema operativo "estilo UNIX", el nombre del sistema a desarrollar y del proyecto encargado de hacerlo no podía ser menos que una referencia al UNIX. Siguiendo una vieja costumbre *hacker,*[4] Stallman decidió llamar "GNU" al sistema operativo, y "Proyecto GNU" al proyecto encargado de llevar a cabo la tarea. La referencia a UNIX estaba en el acrónimo recursivo "GNU", que significa "*GNU's not UNIX*", *es decir, "G(NU) N(o) (es) U(nix)".*

En ese mismo año (1984), Stallman y un grupo de desarrolladores que colaboraban con él desarrollaron dos herramientas fundamentales para un sistema operativo estilo UNIX: el compilador "GCC" para lenguaje C y el editor de texto "EMACS"[5] con "Lips"[6] para escribir comandos de edición.

La envergadura del "Proyecto GNU" induce a Stallman a fundar, en 1985, la *Free Software Foundation (FSF)*, una organización de caridad libre de impuestos para el desarrollo de *software* libre. Esta entidad tendrá, entre otros, el fin de funcionar "como receptora de fondos y recursos que ayuden al desarrollo del proyecto GNU, y como dueña de la propiedad intelectual generada por el proyecto, función, ésta última, por demás extraña, si tenemos en cuenta los esfuerzos que

[4] *HACKERS:* Originariamente, personajes con cierta aureola, capaces de penetrar en bases de datos de centros oficiales, incluso, sin extraer información secreta y sin ánimo de causar daño.

[5] *EMACS:* Es un editor de texto con una gran cantidad de funciones, muy popular entre programadores y usuarios técnicos. GNU Emacs es parte del proyecto GNU, activamente desarrollado. Es la versión más popular de Emacs. El manual de GNU Emacs lo describe como "un editor extensible, personalizable, auto-documentado y de tiempo real." Es la versión más compatible y portada de las implementaciones de Emacs. La última versión, liberada el 2 de junio de 2007, es la 22.1 .El EMACS original significa, *Editor MACroS* para el TECO. Fue escrito en 1975 por Richard Stallman junto con Guy Steele. Fue inspirado por las ideas de TECMAC y TMACS, un par de editores TECO-macro escritos por Guy Steele, Dave Moon, Richard Greenblatt, Charles Frankston y otros

[6] LIPS (Logical Inferences Per Second) Inferencias lógicas por segundo. Unidad que mide la velocidad del pensamiento de una aplicación en inteligencia artificial. Los seres humanos pueden realizar cerca de 2 LIPS. En el computador, un LIPS equivale desde 100 a 1,000 instrucciones.

hacen Stallman y la FSF para desterrar de una vez y para siempre a la "propiedad intelectual" y la mil veces proclamada dicotomía "*software* libre-*software* propietario". En este mismo año, Stallman publica el Manifiesto GNU, que es una suerte de acta fundacional o declaración de principios tanto del proyecto GNU como de la FSF.

Dicho brevemente, el objetivo del proyecto GNU era hacer ese sistema operativo de modo que nadie tuviera que pagar por el *software* y generar una comunidad a partir de él. Stallman lanzó el proyecto GNU porque esencialmente pensaba que el conocimiento que constituye un programa corriente –que la informática llama código fuente– debe ser libre. Si no lo fuera, argumentaba Stallman, sólo muy poca gente con gran poder dominaría la informática. Donde los vendedores comerciales propietarios del *software* vieron una industria que guardaba los secretos comerciales a fin de que fueran protegidos firmemente, Stallman considera que el conocimiento científico debe ser compartido y distribuido.

Al principio el espíritu comunitario parecía bastar por sí mismo para llevar a cabo la tarea. Pero el punto de inflexión se produjo cuando la empresa *Symbolics* solicitó permiso a Stallman para usar el *Lips*. Este prestó su consentimiento a través de una versión libre de la obra. *Symbolics* amplió y mejoró el *software*, y cuando Stallman quiso acceder a esas mejoras, *Symbolics* no se lo permitió. A raíz de este acontecimiento, Stallman procedió a redactar una licencia que permitiese a los usuarios seguir usando, copiando, estudiando, modificando o redistribuyendo el *software* creado y desarrollado por la FSF, pero que les impidiese apropiarse de las modificaciones que en el futuro ellos mismos realizasen, o combinar *software* GNU con otro tipo de *software.*

La licencia, cuya primera versión data de 1989, fue llamada *General Public License (GPL)*, y el principio jurídico que le sirvió de inspiración fue bautizado, siguiendo también una costumbre *hacker, copyleft.*

En 1990 el Proyecto GNU estaba a punto de cumplir su objetivo. El sistema operativo libre estilo UNIX estaba casi completo. Desde sus

comienzos en 1984 hasta la fecha, la FSF y terceros colaboradores habían estado desarrollando las herramientas fundamentales del sistema, reemplazando una a una las del UNIX. Sólo faltaba una pieza clave: el *kernel*[7] o núcleo del sistema. La FSF estaba trabajando sobre el *Kernel* en un proyecto llamado "GNU HURD"[8].

Recuérdese que, por esa época y sin conexión alguna con la FSF y el Proyecto GNU, el CSRG[9] de la Universidad de California también estaba tratando de reescribir el código *kernel,* del UNIX, por los problemas que había con su licencia. Como se evidenció, en 1991 el CSRG liberó la *Networking Release 2* con la mayoría del código del *kernel* reescrito y, tan sólo unos pocos meses después, apareció la 386BSD[10] con todo ese código reescrito.

En julio de 1991, aparece en escena Linus Torvalds, un joven estudiante filandés que, por su propia cuenta y riesgo, puso un mensaje en Internet, por entonces ya bien avanzada, en el que anunciaba su proyecto de hacer un sistema libre para reemplazar a MINIX[11]. De manera caótica y a ritmo vertiginoso, *hackers* de todas partes del

---

[7] *KERNEL:* Núcleo del sistema operativo. Es el que se encarga de las labores de más bajo nivel (el nivel más cercano al *hardware*), tales como gestión de memoria, de entrada/salida de dispositivos, etc. El kernel más popular en el mundo del software libre es Linux, aunque hay muchos más (por ejemplo los sistemas BSD tienen uno propio).

[8] *GNU HURD:* Kernel del proyecto GNU cuya pretensión es sustituir algún día a Linux. Actualmente en desarrollo. Nombre del nucleo del sistema que sigue desarrollando la FSF dentro del proyecto GNU.

[9] *CSRG:* Sobre la misma época (1989), en el Computer Science Research Group (CSRG) de la Universidad de California en Berkeley poseían toda una serie de aplicaciones, desarrolladas con el objetivo de mejorar UNIX, que formaron lo que se conoce como "BSD Unix".

[10] 386BSD: En California, Bill Jolitz estaba implementando las partes que faltaban de la distribución Net/2 (distribución resultado del intento de CSRG de Berkeley de disponer de una versión de BSD Unix libre de licencias propietarias). El resultado de Bill obtuvo el nombre de 386BSD el cual incluía un completo núcleo más diversas utilidades bajo licencia BSD. Más tarde, el código de 386BSD daría lugar a NetBSD, FreeBSD y OpenBSD.

[11] MINIX: Versión de UNIX para computador personal, Mac, Amiga y Atari ST desarrollado por Andrew Tannenbaum, publicado por Prentice-Hall. Viene con un código fuente completo.

mundo, conectados a través del correo electrónico y los grupos de *news*, "van dejando nuevas versiones en los repositorios FTP[12] de Internet para que la gente las pruebe, las estudie o las mejore.

Tan sólo dos meses después se liberó la primera versión del *kernel* (0.01), aunque recién en 1994 apareció la primera versión estable (1.0).

Aunque la obra de Torvalds, según se dijo, fue llevada a cabo sin contacto alguno con la FSF y el Proyecto GNU, significó para éste un aporte importantísimo, porque Linux fue liberado bajo la licencia GPL. Esta decisión de Torvalds posibilitó a la FSF, en 1992, "combinar Linux con el sistema no tan completo de GNU", resultando así "un sistema operativo libre completo […]. Es gracias a Linux que podemos ver funcionar un sistema GNU en la actualidad". Precisamente, esta versión fue denominada "GNU / Linux, para expresar su composición como combinación de un sistema GNU con Linux como núcleo" (Stallman,1998)

b) **La *Open Source Innitiative***

Estamos en la primera mitad de la década de los noventa. La comunidad *hacker* cuenta con dos sistemas operativos completos que los usuarios pueden usar, copiar, estudiar, modificar o distribuir, desarrollado de manera independiente y que, aparentemente, constituyen una interesante alternativa a los sistemas operativos propietarios.

En 1994 GNU / Linux era la única alternativa competitiva frente a los productos propietarios. Pero, según algunos *hackers*, tenía un problema.

En efecto, la prédica libertaria, reaccionaria y ofensiva de Stallman había provocado repulsa en el mundo empresarial hacia el *software*

[12] FTP de Internet: FTP es uno de los diversos protocolos de la red Internet, concretamente significa *File Transfer Protocol* (Protocolo de Transferencia de Archivos) y se utiliza para transferir datos por la red.

libre en general, identificando a éste con la ideología de la FSF. Este mensaje radical (el de Stallman) llevó a muchas compañías de *software* a rechazar absolutamente al *software* libre. Después de todo, ellas están en el negocio de hacer dinero, no en el de proporcionarnos un cuerpo de conocimientos. Para Stallman, esta escisión entre la industria del *software* y la informática era aceptable e, incluso, tal vez deseable. No así para otros *hackers*, que apoyaban al *software* libre y al modelo de producción inaugurado por Tovarlds (modelo bazar), pero no a la ideología de la comunidad que se había formado en torno a éste.

Para ellos el modelo bazar de producción de *software* libre constituía una interesante oportunidad para el mundo empresarial, dado que aparentemente permitía el desarrollo de *software* de alta calidad en períodos de tiempo más cortos que los empleados en el mundo del *software* propietario y aun costo mucho más bajo.

El modelo bazar de desarrollo de *software*, que utiliza lo que Raymond, uno de los miembros fundadores de la *open source software,* llama la ley de Linux (Raymond, www.catb.org), puede resumirse en una serie de consignas y principios que fueron utilizados por Linus Torvalds al desarrollar el *kernel* del sistema GNU/Linux, que luego liberaría bajo licencia GPL.

Raymond explica que el *kernel* de Torvalds surgió "como por arte de magia, gracias a la actividad *hacker* desplegada en ratos libres por varios miles de programadores diseminados en todo el planeta, conectados solamente por los tenues hilos de la Internet".

Raymond era un ferviente admirador del *software* libre. De hecho, había realizado muchos aportes al movimiento. Sin embargo, creía que determinada complejidad crítica del desarrollo de *software* exigía que esto se hiciera centralizando determinadas decisiones en genios trabajando encerrados, sin liberar versiones antes de tiempo. Para él, Linus Torvalds demostró que era mejor el modelo contrario, es decir, libre rápido y a menudo abierto.

Con estos antecedentes y preocupados por la resistencia que el *software* libre tenía en la comunidad empresarial, en 1997 se reúne un grupo

de líderes del movimiento con el objetivo explícito de "encontrar una manera de promover las ideas que rodeaban al *software* libre a la gente que anteriormente había rechazado el concepto. Acordaron que el mensaje anticomercial de la FSF había impedido al mundo apreciar realmente el poder del *software* libre. Integraban el grupo entre otros Raymond y Perens

La conclusión a la que arribaron fue que lo que les estaba faltando era una campaña de *marketing*, no sólo para ganar una parte del mercado informático, sino también una parte de la opinión pública. A estos fines propusieron *open source software* como nueva expresión para designar al software libre, porque además de evitar la ambigüedad del término inglés *free* (libre y gratis), permitía diferenciarlo de las ideas y de la comunidad de Stallman.

En el mismo año 1997, Raymond contactó a Perens, por ese entonces líder del proyecto Debian, una distribución de GNU/Linux iniciada en 1993, con la idea de fundar una entidad similar a la FSF, pero con el objetivo de promocionar la campaña *open source software*. El propio Perens y Raymond fundaron la *Open Source Innitiative (OSI)*, entidad que vio luz en 1998.

Se puede apreciar también que en los inicios de 1998, *Netscape* decidió liberar su producto, el *Navigator 5.0*.

Esta decisión de *Netscape* significó la primera batalla ganada por el movimiento, pero fue un triunfo fundamental: "el anuncio de *Netscape* de liberar el código de su *navegator* supuso un fuerte impacto en la industria del *software*. Muchas empresas comenzaron a considerar el *software* libre como algo digno de estudio, y algo que podría interesarles mucho. También los mercados financieros se empezaron a ocupar del *software* libre", y en septiembre de 1998, *Red Hat Linux*, una empresa distribuidora de GNU/Linux, "anunció que *Netscape e Intel* habían invertido en ella" (Raymond, 1997).

Son momentos de euforia para los partidarios del *software* libre. A partir de entonces, la brecha abierta por Stallman y la FSF entre el mundo empresarial y el *software* libre comenzó a acortarse.

SIGNIFICADO Y DEFINICIÓN DEL *SOFTWARE* LIBRE

El *software* libre es aquel que puede ser distribuido, modificado, copiado y usado; por lo tanto, debe venir acompañado del código fuente para hacer efectivas las libertades que lo caracterizan. Dentro de *software* libre hay, a su vez, matices que es necesario tener en cuenta. Por ejemplo, el *software* de dominio público significa que no está protegido por el *copyright*, por lo tanto, podrían generarse versiones no libres del mismo, en cambio el *software* libre protegido con *copyleft* impide a los redistribuidores incluir algún tipo de restricción a las libertades propias del *software* así concebido, es decir, garantiza que las modificaciones seguirán siendo *software* libre, y es sobre este tipo de *software* sobre el cual el presente trabajo hace referencia, es decir, sobre el *software* libre protegido con *copyleft*. También es conveniente no confundir el *software* libre con el *software* gratuito, este último no cuesta nada, hecho que no lo convierte en *software* libre, porque no es una cuestión de precio, sino de libertades. Para Richard Stallman el *software* libre es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en "libertad de expresión". En términos del citado autor el *software* libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*. Y se refiere especialmente a cuatro clases de libertad para los usuarios de *software*, a las que ya se hizo referencia.

De modo que los usuarios, sean desarrolladores o usuarios sin conocimiento tecnológico deberían ser libre de redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar. Gozar de esta libertad significa, entre otras cosas, no tener que pedir permiso ni pagar para ello. Asimismo, debería ser libre para introducir modificaciones y utilizarlas de forma privada, ya sea en su trabajo o en su tiempo libre, sin siquiera tener que mencionar su existencia. Si se decidiera publicar estos cambios, no se debería estar obligado a notificárselo a ninguna persona ni de ninguna forma en particular. La libertad para utilizar un programa significa que cualquier individuo u organización podrán ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna

entidad en concreto. Esto es parte del derecho a disponer de la obra que tienen los autores de la misma. La libertad para redistribuir copias supone incluir las formas binarias o ejecutables del programa y el código fuente tanto de las versiones modificadas, como de las originales, ya que debemos tener la libertad para redistribuir tales formas si se encuentra el modo de hacerlo, pues las libertades para hacer cambios y para publicar las versiones mejoradas requieren de la accesibilidad de código fuente, por supuesto de manera libre, condición necesaria del *software* libre. Cuando hablamos de *software* libre, debemos evitar utilizar expresiones como "regalar" o "gratis", ya que se puede caer en el error de interpretarlo como una mera cuestión de precio y no de libertad (Stallman, 2004).

CATEGORÍAS DE *SOFTWARE* LIBRE Y NO LIBRE

Desde el punto de vista de las libertades ofrecidas a los usuarios, la FSF, ha presentado la siguiente categorización:

Puede observarse como el criterio de la libertad, divide al "mundo del *software*" en dos grandes categorías o géneros: software libre (*free software)* y el *software* de fuente abierta (*open source software*). Por otro lado, aparece el *software* propietario (*propietary software).*

Dentro del primer grupo figuran tres subcategorías o especies: el *software* de dominio público (*public domain software*), el *software* no protegido con *copyleft* (XFree86 Style, que es lo mismo que licencias tipo BSD) y el *software* protegido con *copyleft (copylefted),* dentro de la cual, a su vez, se encuentra el *software* licenciado bajo los términos de la GNU GPL.

a) *Software* de dominio público

Este tipo de *software* es una de las categorías fundamentales de software. De hecho, para Stallman y para la FSF, es una subcategoría de *software* libre. "El *software* de dominio público,… es un caso especial de software libre no protegido con *copyleft…"*, "el modo más simple de hacer un programa libre es ponerlo a dominio público…", "el *software* de dominio público…es *software* libre…" (FSF, 1999). Y, según

Carranza (2004), esta postura errónea de Stallman y de la FSF no es casual. Y bien, ¿qué es el *software* de dominio público?

Para comprender acabadamente este punto conviene explicar qué es el dominio público en general. Es conveniente volver a las nociones básicas de propiedad intelectual y derechos de autor. Los derechos de propiedad intelectual, que son aquellos que se confieren a las personas sobre las creaciones de su mente, se dividen en dos categorías, a saber: la propiedad industrial y los derechos de autor, sistema este último bajo el que se encuentra protegido el *software*.

Los derechos de propiedad intelectual tienen dos caracteres comunes: su objeto lo constituyen bienes inmateriales y conceden a sus titulares derecho exclusivo (monopolio) sobre la utilización de sus creaciones, derecho que, por eso mismo, es oponible *erga omnes*.

A fin de equilibrar los derechos morales y económicos de los creadores respecto de sus creaciones, y los derechos del público para acceder a esas creaciones, las leyes de propiedad intelectual limitan esa exclusividad de dos maneras: concediéndola sólo por un plazo determinado y sujetándola a una serie de limitaciones y excepciones, como, por ejemplo, el derecho de uso para fines didácticos o científicos.

Cuando una obra intelectual cualquiera, cae bajo este régimen, se dice que está en el dominio privado, que una persona determinada, física o jurídica, privada o pública, tiene actualmente la titularidad de esos derechos morales y patrimoniales. Toda obra, entonces, cuyo dueño es alguien determinado, está en el dominio privado, y esto quiere decir que sólo esa persona determinada tiene derecho a utilizarla, y que los demás tendrán ese derecho solamente en el caso de que ella lo autorice. El derecho del propietario de la obra es exclusivo, en el sentido de que los que no son propietarios están en principio, excluidos de su utilización y que aquél tiene derecho a excluirlos.

"El titular del derecho de autor de una obra protegida puede usar la obra como él lo desee y puede excluir a otros del uso no autorizado

de ella. Por lo tanto, los derechos reconocidos por la ley al titular del derecho de autor de una obra protegida son descritos frecuentemente como "derechos exclusivos" para autorizar a otros a usar la obra protegida" (OMPI www.wipo.int Consultado: 10-01-08)

Ahora bien, el dominio público se diferencia en primer lugar, del dominio privado; porque una obra que pertenece al dominio público no pertenece al dominio privado, es decir, no tiene un dueño determinado. Por consiguiente, el dominio público implica, a diferencia del dominio privado, ausencia de exclusividad, en el sentido de que ninguna persona determinada tiene, respecto de una obra perteneciente al dominio público, un derecho exclusivo de utilización "internacionalmente el dominio público es el conjunto de obras creativas y de otros conocimientos –escritos, obras de arte, música ciencia, invenciones y otros– respecto de los cuales ninguna persona u organización tienen intereses propietarios (típicamente un monopolio garantizado por el Gobierno, tal como el *copyright* o derecho de autor o una patente).

Pero el dominio público también debe ser distinguido de lo que se denomina *res nullius* (cosa de nadie). *Res nullius* alude a los bienes que no tienen dueño, no ya que no tienen como dueño a una persona determinada, sino que no tienen ningún tipo de dueño. Precisamente porque no tienen ningún tipo de dueño, cualquiera puede adquirir su propiedad, generalmente a través de un modo de adquisición de la propiedad conocido con el nombre de "apropiación". En rigor, la categoría *res nullius* no puede ser aplicada a las obras intelectuales, porque éstas, desde el momento mismo de su creación y sin necesidad de cumplir formalidad alguna, tienen dueño. (art. 5.3 Convenio de Berna para la Protección de las Obras Literarias y Artísticas)

En el caso de las obras de dominio público podemos decir que el dueño de una obra que está en el dominio público somos todos, que es la sociedad en su conjunto y que, por eso mismo, no lo es ninguno de sus componentes individualmente considerados. Por esta razón esas obras pueden ser utilizadas por cualquiera sin necesidad de permiso alguno.

Entonces, según la FSF "el *software* de dominio público es *software* que no está protegido con *copyright*… "dominio público" es un término legal y significa de manera precisa "sin *copyright*". Está claro, entonces, que de dominio público es todo *software* que no está protegido por las leyes de derecho de autor, y esto quiere decir que ninguna persona determinada tiene la titularidad de los derechos patrimoniales que esas leyes reconocen y que, por consiguiente, ninguna goza ni puede gozar de un derecho exclusivo sobre su utilización. Pero esto no significa que cualquiera pueda apropiarse del *software* que está en el dominio público. Basta sólo recordar que dominio público no es lo mismo que *res nullius.* "Pueden ser usadas… por cualquier persona sin que ninguna pueda adquirir derechos exclusivos sobre ella: sí en cambio, sobre los aportes creativos que se le adicionen… o sobre las nuevas obras resultantes" (Lipszyc, 1993)

Ante lo anterior, surgen algunas interrogantes: ¿qué sucede si una persona modifica ese tipo de *software* o crea a partir de él una obra derivada? Esa modificación u obra nueva, ¿también va estar en el dominio público? La respuesta es No. Esa modificación u obra derivada va a ser propiedad del programador o desarrollador que la haga, y él va a ser el único titular de la obra y va a gozar de un derecho exclusivo sobre su utilización. ¿Por qué? Porque las obras derivadas son obras distintas de las originarias.

b) *Software* libre y *software* de fuente abierta

Los términos *free software y open source software* no son usados para dar a entender más o menos lo mismo, sino, más bien, para dar a entender ideologías distintas. No obstante, dado que se ocupan de la misma realidad, a saber: el llamado *free software u open source software*, definen esa realidad de manera análoga. Precisamente, dada esta analogía y siguiendo la propuesta de Carranza (2004), usaremos la expresión "f/oss" para referirnos, de ahora en adelante, a la realidad a las que esas definiciones hacen referencia; es decir, tanto al *software* libre de Stallman, como al *software* de código abierto de Raymond y Pernes.

Por su parte, Webbink (2003) afirma que "comparando las definiciones *open source y free software,* uno encuentra que todo el *Free Software* es O*pen Source*, pero, tal como es administrado por la *Free Software Foundation*, no todo el *open source* es *free software*"

Ya se ha mencionado que el *software* libre se refiere a cuatro libertades para los usuarios:

1. Libertad de usar el programa, con cualquier propósito, sobre esta libertad, la FSF mantiene que el término "usar", referido a un programa, comprende cualquier tipo de uso, en cualquier sistema informático, para cualquier clase de trabajo y sin estar obligado a comunicárselo a nadie.

2. Libertad de estudiar cómo funciona el programa, y adaptarlo a las necesidades. El acceso al código fuente es una condición previa para esto. Acerca de esta libertad, especifica la FSF que los usuarios deben poder modificar el programa según sus necesidades o conveniencias y utilizar esas modificaciones de manera privada "sin siquiera tener que anunciar que dichas modificaciones existen". Es más, si un usuario publica esas modificaciones, no tiene "por qué avisar a nadie en particular, ni de ninguna manera en particular".

3. Libertad de distribuir copias, con lo que puedes ayudar a tu vecino y

4. Libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

De lo anterior, de acuerdo con las libertades 3 y 4, afirma la FSF que "deberías tener la libertad de distribuir copias, sea con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y a cualquier lugar… La libertad de distribuir copias debe incluir tanto las formas binarias o ejecutables del programa como su código fuente, sean versiones modificadas o sin modificar".

También se debe aclarar con relación a la libertad 4, que la FSF, habla, impropiamente de "libertad de mejorar el programa". En realidad, es mejor hablar de "libertad de modificar el programa", porque un programa puede ser, ciertamente, modificado para bien, en cuyo caso estaremos frente a una verdadera "mejora", pero no es menos cierto que también puede ser modificado para mal, caso en el que, no se estará frente a una mejora, sino más bien ante un "empeoramiento"

El término "mejora" tiene una carga valorativa, mientras "modificación" es meramente descriptivo.

Según la FSF, condiciones necesarias para que estas "libertades" tengan sentido son las siguientes:

1. Que se pueda acceder al código fuente del programa.
2. Que ellas sean irrevocables mientras el usuario no haga nada incorrecto, lo cual significa, según los términos de la GNU GPL, no violar los términos de la propia licencia. "Si el desarrollador del software tiene el poder de revocar la licencia aunque no le hayas dado motivos, el *software* no es libre".

Finalmente, la FSF insiste en que justamente porque "el *software* libre" es un asunto de libertad, no de precio y el ser libre de hacer esto significa (entre otras cosas) que no tienes que pedir o pagar permisos, software libre no significa no comercial.

c) **F/OSS protegido y no protegido con *Copyleft***

¿Qué es proteger con *copyleft?*

Evidentemente, la noción central al respecto es la de *copyleft.* No se puede entender qué son el f/oss protegido y el f/oss no protegido con *copyleft*, si no se comprende previamente este último concepto.

El *copyleft* es una de las estrategias jurídicas empleadas por la comunidad F/OSS para, haciendo uso de las leyes de derecho de autor,

evitar sus consecuencias respecto de la "libertad". Pero, por ahora sólo vamos a centrarnos en el concepto de *copyleft*.

Volvamos a la noción –por un momento– de dominio público, que ya nos es familiar y, procuraremos analizar desde ella la de *copyleft*.

Recién recordemos que Stallman y la FSF evitan poner en el dominio público el *software* que producen. ¿Por qué? Porque, conforme a la FSF, si bien este régimen da lugar a "que la gente comparta el programa y sus mejoras, si así lo desean", también permite "a quien no quiera cooperar convertir el programa en *software* propietario. Pueden hacer cambios y distribuir el resultado como un producto propietario…", como consecuencia de lo cual, "las personas que reciban el programa en su forma modificada no poseen la libertad que el autor original les dio debido a que el intermediario se la ha retirado".

Ahora bien, como es intención de Proyecto GNU "dar a todos los usuarios la libertad de redistribuir y cambiar *software* GNU… en vez de poner *software* GNU bajo dominio público, lo hacemos *copyleft"*. "La meta de GNU era dar libertad a los usuarios, no solo ser popular. Por lo tanto, debíamos usar términos de distribución que impidieran que el *software* GNU se transformara en *software* propietario. El método que utilizamos se denomina *copyleft"* (Stalman ,1998).

Estas afirmaciones bastan para situar adecuadamente al *copyleft* en relación con el dominio público.

Para Stallman y la FSF, el régimen del dominio público tiene una característica positiva y una negativa.

La positiva es que permite a la gente compartir el *software* y sus mejoras, si así lo desean. En efecto, el *software* que está en el dominio público no está protegido por las leyes de derecho de autor, y esto quiere decir que ninguna persona determinada tiene la titularidad de los derechos patrimoniales que esas leyes reconocen y que, por consiguiente, ninguna goza ni puede gozar de un derecho exclusivo sobre su utilización. Como contrapartida, cualquiera puede utilizar ese tipo de *software* de la manera en que lo crea conveniente.

La negativa es que posibilita convertir al *software* que está bajo ese régimen en *software* propietario. Los usuarios pueden hacer cambios –obra derivada– y distribuir el resultado de esos cambios como un producto propietario, según se indicó, las obras que están en dominio público pueden ser "usadas…" por cualquier persona sin que ninguna pueda adquirir derechos exclusivos sobre ella; sí, en cambio, sobre los aportes creativos que se le adicione… o sobre las nuevas obras resultantes". Esto conlleva, en opinión de Stallman y de la FSF, a que "las personas que reciban el programa en su forma modificada no posee la libertad que el autor original les dio debido a que el intermediario se la ha retirado".

Dicho de otro modo, el régimen de dominio público implica, ciertamente, las facultades de usar, copiar, estudiar, modificar o distribuir el *software* tal como está, pero también supone la facultad de distribuirlo bajo la forma propietaria.

Para Stallman y la FSF, el *copyleft* es, por decirlo de alguna manera, el régimen jurídico que, salvaguardando la característica positiva del dominio público, evita su aspecto negativo. El *copyleft* es, entonces, una regla relativa a la libertad 4: "libertad de distribuir copias de f/oss, que impide que los usuarios de f/oss puedan hacer cambios a ese tipo de *software* y distribuir el resultado de esos cambios como un producto propietario. Y esta regla evita, aparentemente el aspecto negativo del régimen de dominio público.

Pero ¿cómo salvaguarda el *copyleft* su aspecto positivo?, ¿cómo asegura que cualquiera puede utilizar f/oss de la manera en que lo crea conveniente sin tener que pedir permiso por ello ni pagar derechos de autor? Conforme a Stallman y a la FSF, conservando el autor la titularidad de los derechos morales y patrimoniales que por ley le corresponden y renunciando a la exclusividad de estos últimos a través de mecanismos específicos de licenciamiento, tales como la GNU GPL.

De lo anterior, se tiene que, a diferencia del dominio público, el *copyleft* no implica "no protegido con derecho de autor", "sin derechos de

autor" o "fuera del alcance de las leyes de derecho de autor": Al contrario, bajo este régimen jurídico una persona determinada tiene la titularidad de los derechos de autor y goza, en consecuencia, de un derecho exclusivo a la utilización del *software*. Y en este sentido, el *copyleft* no se diferencia en nada del sistema de *copyright o derechos de autor.*

Pero, lo que otorga su especificidad al *copyleft* es la manera en que ese titular determinado ejerce los derechos que la ley le reconoce.

En efecto, el *copyleft* se da cuando, en vez de conservar y ejercer el monopolio de explotación que legalmente le corresponde, el titular de los derechos de autor renuncia a esa exclusividad, pero lo hace bajo la condición de que las futuras distribuciones de su *software*, en su versión original o en versiones modificadas, concedan a los potenciales usuarios las mismas facultades de utilización que él confirió.

En otras palabras, el *copyleft* implica que el titular de los derechos de autor renuncia, respecto de potenciales usuarios, al derecho de exclusividad sobre la utilización de su *software*, pero esa renuncia está sujeta a una condición resolutoria, a saber, que esos potenciales usuarios agreguen restricciones al momento de distribuir el *software*.

La idea central del *copyleft* es que le damos a cualquiera el permiso para correr el programa, copiar el programa, modificar el programa y redistribuir versiones modificadas –pero no le damos permiso para agregar restricciones propias–. De esta manera, las libertades cruciales que definen al "*software* libre" quedan garantizadas para cualquiera que tenga una copia: se transforman en derechos inalienables. Para que el *copyleft* sea efectivo, las versiones modificadas deben también ser libres. Es por eso que la FSF termina afirmando que *copyleft* es la forma general de hacer un programa *software* libre y requiere que todas las modificaciones y versiones extendidas del programa sean también *software* libre.

Ahora sí, ¿Qué es el f/oss protegido con *copyleft*? Es *software* libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el *software*. Esto significa que cada copia del *software*, aun si ha sido modificado, debe ser *software* libre. Ejemplo de este tipo de f/oss es el software licenciado bajo los términos de la GNU GPL.

¿Qué es el f/oss no protegido? Es el *software* libre que viene desde el autor con autorización para redistribuir y modificar así como para añadirle restricciones adicionales. Si un programa es libre, pero no protegido con *copyleft*, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de *software* puede compilar el programa, con o sin modificaciones y distribuir el archivo ejecutable como un producto propietario de *software*. Ejemplo de esta especie de f/oss es el *software* licenciado en los términos BSD.

Pero entonces, ¿en qué se distinguen el f/oss no protegido con *copyleft* y el *software* de dominio público? Entre un sistema y otro hay una diferencia fundamental, a saber, que el *software* de dominio público está fuera del alcance de las normas de derechos de autor, mientras que el f/oss no protegido con *copyleft* es una manera de ejercer los derechos de autor.

El *software* de dominio público no tiene como propietario a una persona determinada, sino a la sociedad en su conjunto. En consecuencia, ninguna persona determinada goza de exclusividad respecto de su utilización, sino que todos, tienen en virtud de la ley, el mismo derecho con utilizarla.

En cambio, el f/oss no protegido con *copyleft* tiene como propietario a una persona determinada, que goza, legalmente, de un derecho exclusivo a su utilización, y si todas las demás personas tienen igual derecho a utilizar ese *software,* es solo en virtud del hecho de que ese titular determinado ha renunciado, a través de una licencia, a esa exclusividad.

d) ***Software* comercial y gratuito**

El *software* comercial es aquel que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del *software*. "Comercial" y "propietario", no son la misma cosa, la mayoría del *software* comercial es propietario, pero hay *software* libre comercial y hay *software* no libre no comercial.

No debe usarse "comercial" como sinónimo de "no Libre", se estarían confundiendo los términos. Un programa es comercial si se desarrolla como parte de una actividad económica. Un programa comercial puede ser libre o no libre, según su licencia. El *software* libre comercial es una contribución a la comunidad f/oss, por lo que debe ser promovido. Reacuérdese que el *software* libre es una cuestión de libertad, no de precio.

2. ASPECTOS JURÍDICOS DEL *SOFTWARE* LIBRE: PROTECCIÓN Y LICENCIAMIENTO

2.1. Un programa libre supone un modo particular de ejercicio del derecho de autor

De acuerdo con la Organización de la Propiedad Intelectual (OMPI), los programas libres suponen una forma de ejercicio del derecho de autor. El *software* libre se define en función de unas libertades del usuario que corresponden a un ámbito previo de derechos del titular, legalmente establecido, y cuyo uso se cede mediante licencia.

Se trata, entre otros, de los derechos de reproducción y distribución del programa, así como los que permiten su transformación.

Junto a estas libertades el usuario puede quedar sujeto a una serie de restricciones que derivan también del ámbito de los derechos del titular.

Algunas de estas restricciones a las libertades –como la atribución de la autoría– pueden derivar no sólo de las condiciones de la licencia,

sino directamente de un derecho irrenunciable e intransferible que la legislación establezca frente a todos, más allá de una relación contractual, p.ej. el derecho moral de paternidad.

Copyright versus Copyleft. Alcances e implicaciones

a) El contrato de *copyright*

El sistema de *copyright* funciona mediante la concesión de privilegios, y por lo tanto de beneficios, a los editores y a los autores, pero no lo hace en su provecho. Más bien lo hace para modificar su comportamiento: proporciona un incentivo a los autores para escribir y editar más. En la práctica, el gobierno emplea los derechos naturales del público, en nombre del público, como parte de un trato para ofrecerle un mayor número de obras editadas. Los expertos en derecho llaman a este concepto el (contrato de *copyright*); como la adquisición estatal de una autopista o un avión usando el dinero de los contribuyentes, excepto que en este caso el gobierno gasta nuestra libertad en lugar de nuestro dinero. Pero ¿tal y como existe en la actualidad, el contrato supone un buen trato para el público? Son posibles muchos contratos alternativos; ¿cuál es el mejor? Cualquier medida en relación a la política de *copyrigh*t es parte de esta cuestión. Normalmente, los autores se los ceden a los editores; son los editores, y no los autores, quienes suelen ejercer los derechos y quienes se quedan con la mayoría de los beneficios, aunque los autores consigan una pequeña porción. Por eso los editores son los que con frecuencia presionan más para aumentar los poderes del *copyright*. Para reflejar mejor la realidad del *copyright* en lugar del mito, este artículo se refiere a los editores antes que a los autores como sujetos de los derechos del *copyright.* También se refiere a los usuarios de las obras protegidas por el *copyright* como "lectores", aún cuando este uso no siempre signifique su lectura, en la medida en que el término "usuario" resulta lejano y abstracto.

¿Cuál es la manera adecuada de decidir la política de *copyright*? Si el copyright es un contrato hecho en nombre del público, debería servir ante todo al interés público. Dado que no podemos encontrar este precio mínimo en términos de libertad a través de un concurso público

como se hace con los proyectos de construcción, ¿qué podemos hacer? Un método posible es reducir los privilegios del *copyright* por etapas y observar los resultados. Observando si hay disminuciones apreciables en el volumen de publicación, aprenderemos qué extensión del *copyright* es realmente necesaria para llevar a cabo los propósitos del público. Esto se debe valorar por medio de la observación práctica, no por lo que los editores digan que ocurrirá, ya que tienen todos los motivos para hacer predicciones exageradas de pérdidas si sus poderes se ven reducidos de algún modo. La política de *copyright* tiene varias dimensiones independientes que pueden ajustarse de forma separada. Después de encontrar el mínimo necesario para una vertiente de esta política, todavía sería posible reducir otras dimensiones del *copyright* a la vez que se mantiene el nivel de publicación deseado. Una dimensión importante del *copyright* es su duración, que ahora se encuentra en torno a un siglo de media. Reducir el monopolio sobre la copia a diez años, desde la fecha en que una obra es publicada, sería un buen primer paso. ¿Por qué diez años? Porque esta es una propuesta segura; podemos confiar en el terreno práctico que esta reducción tendrá, hoy en día, poco impacto en la viabilidad general de la edición. En muchos medios y géneros, las obras de éxito son muy rentables unos pocos años, y normalmente incluso las obras de éxito ya no se editan pasados los diez años.

En el campo de la programación informática, tres años deberían bastar, dado que los ciclos de un producto son incluso más cortos. Otra dimensión de la política de *copyright* es la magnitud del uso razonable: algunas formas legalmente permitidas de reproducción de un trabajo, total o parcialmente, aún cuando éste está protegido por el *copyright*. El primer paso natural para reducir este aspecto del *copyright* es permitir la copia privada sin ánimo de lucro, ocasional y en pequeña cantidad, para su distribución entre individuos. Esto eliminaría la intrusión de la policía del *copyright* en la vida privada de la gente, pero probablemente tendría poco efecto en las ventas de las obras publicadas. Para las novelas, y en general para las obras que se utilizan como entretenimiento, la redistribución textual no comercial podría ser una libertad suficiente para los lectores. Los programas informáticos, al ser usados para fines funcionales como para trabajar, exigen libertades adicionales, incluyendo la libertad de publicar una versión mejorada. Sin embargo,

en relación a estas libertades un compromiso aceptable podría ser que estuvieran disponibles universalmente únicamente después de una retraso de dos o tres años con respecto a la publicación del programa. Al igual que consideramos la reducción de la extensión del *copyright*, debemos asegurarnos de que simplemente las empresas mediáticas no lo reemplazarán con acuerdos de licencia para el usuario final.

b) Ventajas del *copyright*

Si no se utilizara *copyright* muchos artistas podrían temer que su trabajo pudiera ser copiado y modificado sin reconocer el trabajo al artista inicial. Sin embargo, esto puede traer problemas: el trabajo del artista podría utilizarse de manera contraria a su voluntad, poniendo una fotografía estándar en un cartel racista. Si el artista es reconocido, será entonces asociado aparentemente con un grupo y una ideología que tal vez no comparta. Asimismo, tampoco hay garantía de que se le atribuya el mérito de su trabajo.

El dueño o la persona que ha escrito una obra, tiene derecho a cobrar por la misma, así como por sus reproducciones.

Las compañías que distribuyen *software* propietario responden ante cualquier problema legal que se suscite respecto a posibles reclamos de propiedad intelectual.

Frente a problemas con los programas de los cuales se han comprado sus licencias propietarias, existe un responsable frente al cual se puede fincar alguna acción legal.

c) Desventajas del *copyright*

Origina que en el mercado se generen monopolios, ejemplo de esto es el claro desarrollo de Microsoft. Los distribuidores de programas generan mercados cautivos, ya que insertan problemas a sus propios programas a efecto de que se requiera alguna actualización.

Debido al constante cambio tecnológico, los programas se vuelven obsoletos sumamente rápido por lo que es necesario comprar las nuevas

versiones de los programas, o bien, los nuevos programas que salgan al mercado; por lo que implica un gran gasto para los usuarios. Debido a las prácticas monopólicas que se ejercen actualmente respecto a los programas propietarios, las compañías se aprovechan de ello y fijan altos costos para sus productos, que en muchos casos son inaccesibles para muchas personas.

Los altos costos derivados de las prácticas monopólicas del *software* propietario han contribuido a la proliferación de la piratería.

d) *Copyleft*

El símbolo del *copyleft* es "", es decir, el símbolo del *copyright* invertido, viendo hacia la izquierda. Es utilizado como la contrapartida del símbolo del *copyright*, sin embargo no posee reconocimiento legal.

El término *copyleft* describe un grupo de licencias que se aplican a una diversidad de trabajos, tales como el *software*, la literatura, la música y el arte. Una licencia *copyleft* se basa en las normas sobre el derecho de autor, las cuales son vistas por los defensores del *copyleft* como una manera de restringir el derecho de hacer y redistribuir copias de un trabajo determinado, para garantizar que cada persona que recibe una copia o una versión derivada de un trabajo, pueda a su vez usar, modificar y redistribuir tanto el propio trabajo como las versiones derivadas del mismo. Así, y en un entorno no legal, el *copyleft* puede considerarse como opuesto al *copyright*. Los vocablos ingleses "*right"* y "*left*" además significan "derecha" e "izquierda" respectivamente, lo que acentúa la diferencia entre ambos conceptos. Una posible traducción al español sería "izquierdos de autor", en contraste con los derechos de autor. En la práctica, sin embargo, el término se deja sin traducir.

e) Métodos de aplicar *copyleft*

La práctica habitual para conseguir este objetivo de explotación sin trabas, copia y distribución de una creación o de un trabajo y sus derivados es la de distribuirlo junto con una licencia. Dicha licencia debería estipular que cada propietario de una copia del trabajo pudiera:

1. Usarla sin ninguna limitación.
2. (Re)distribuir cuantas copias desee, y
3. Modificarla de la manera que crea conveniente.

Estas tres libertades, sin embargo, no son suficientes aún para asegurar que un trabajo derivado de una creación sea distribuido bajo las mismas condiciones no restrictivas: con este fin, la licencia debe asegurar que el propietario del trabajo derivado lo distribuirá bajo el mismo tipo de licencia.

Otras condiciones de licencia adicionales que podrían evitar posibles impedimentos al uso sin trabas, distribución y modificación del trabajo incluirían: asegurar que las condiciones de la licencia *copyleft* no pueden ser revocadas; asegurar que el trabajo y sus derivados son siempre puestos a disposición de manera que se facilite su modificación, para el *software*, esta facilidad suele asociarse a la disponibilidad del código fuente, donde incluso la compilación de dicho código debería permitirse sin ninguna clase de impedimento, idear un sistema más o menos obligatorio para documentar adecuadamente la creación y sus modificaciones, por medio de manuales de usuario, descripciones, etc.

En la práctica, para que estas licencias *copyleft* tuvieran algún tipo de efecto, necesitarían hacer un uso creativo de las reglas y leyes que rigen la propiedad intelectual, cuando nos referimos a las leyes del *copyright*, todas las personas que de alguna manera han contribuido al trabajo con *copyleft* se convertirían en titulares de los derechos de autor, pero, al mismo tiempo, si nos atenemos a la licencia, también renunciarían deliberadamente a algunos de los derechos que normalmente se derivan de los derechos de autor, por ejemplo, el derecho a ser el único distribuidor de las copias del trabajo. Aunque depende de las leyes que rigen la propiedad intelectual, que pueden ser diferentes de un país a otro, la licencia final, que no es más que un método para alcanzar los objetivos del *copyleft*, también puede diferir de un país a otro (Stallman,2004).

f) Ventajas del *Copyleft*

Cuando el *copyleft* rige un trabajo su eficiencia hace cumplir las condiciones de la licencia a todos los tipos de trabajos derivados.

Este tipo de licencias es el que se utiliza generalmente para la creación de bibliotecas de *software*, con el fin de permitir que otros programas puedan enlazar con ellas y ser redistribuidos, sin el requerimiento legal de tener que hacerlo bajo la nueva licencia *copyleft*.

El *copyleft* es aquel que permite que todas las partes de un trabajo (excepto la licencia) sean modificadas por sus sucesivos autores.

El *copyleft* también ha inspirado a las artes, con movimientos emergentes como la "*Free Society*" y los sellos discográficos *open-source*. Por ejemplo, la Licencia *Free Art* es una licencia *copyleft* que puede ser aplicada a cualquier obra de arte.

Han inspirado también la creación de las licencias *Creative Commons* "compartir igual" y la Licencia de Documentación Libre de GNU.

g) Desventajas del *copyleft*

El *copyleft* hace referencia a las licencias que no se heredan a todos los trabajos derivados, dependiendo a menudo de la manera en que éstos se hayan derivado.

Se requiere distribuir los cambios sobre el *software* con "*copyleft*", pero no los cambios sobre el *software* que enlaza con él. Esto permite a programas con cualquier licencia ser compilados y enlazados con bibliotecas con *copyleft*, tales como glibc (una biblioteca estándar requerida por muchos programas) y ser redistribuidos después sin necesidad de cambiar la licencia.

El *copyleft* parcial implica que algunas partes de la propia creación no están expuestas a su modificación ilimitada, o visto de otro modo, que no están completamente sujetas a todos los principios del *copyleft*.

El *copyleft* es más difícil de poner en práctica en aquellas artes que se caracterizan por la producción de objetos únicos, que no pueden ser copiados (a menos que no se tema por la integridad del trabajo original). Se puede ilustrar esta idea con el siguiente ejemplo: suponga que hay

una exposición pública de algunos cuadros mundialmente famosos, algunas de las muchas copias y trabajos derivados que *Andy Warhol* hizo de sus propias obras de arte, y suponga que alguien que tiene acceso a esos cuadros (sin tener plena propiedad de los derechos de éstos), decide "mejorarlos" con algunos efectos pictóricos de su gusto (sin olvidar la correspondiente firma con pintura de spray). Dada esta situación, no habría manera (legal) de detener a este tipo si le puede considerar el titular bajo *copyleft* de dichas obras.

Por otra parte, un programa libre puede licenciarse de varias maneras que reflejan otros tantos modos de ejercicio de los derechos de autor:

2.2. **La Licencia Pública General de GNU (GNU GPL)**

La *Licencia Pública General* (más conocida por su acrónimo en inglés GPL) es con diferencia la licencia más conocida de todas las licencias del mundo del *software* libre. Su autoría corresponde a la *Free Software Foundation* y es también la licencia más utilizada (más del 70% de los proyectos), incluso por proyectos con tanta reputación del mundo del *software* libre y código de fuente abierto como Linux.

La licencia GPL pretende garantizar la libertad de compartir y modificar *software* libre más allá del ámbito contractual inmediato, asegurando que el *software* sea libre también para otros usuarios posteriores.

Las Licencias Públicas Generales están diseñadas para asegurar que quienes contraten en el futuro con el actual licenciatario gocen también de la libertad de distribuir copias de *software* libre (y cobrar por ese servicio si quieren), que reciban el código fuente y puedan modificar el *software* o usar fragmentos de él en nuevos programas libres. Para conseguirlo se prohíbe que el licenciatario pueda negar tales derechos a la persona con la que a su vez contrate

Una licencia es incompatible con la GPL cuando restringe alguno de los derechos que la GPL garantiza, ya sea explícitamente contradiciendo alguna cláusula, ya implícitamente, imponiendo alguna nueva.

Un ejemplo de empresa que basa su negocio en este tipo de licencia sería Ximian, cuyos productos se han distribuido bajo licencias de GNU, tratando de rentabilizarlos consiguiendo contratos para hacerlos evolucionar en ciertos sentidos, para adaptarlos a las necesidades de sus clientes, y ofreciendo personalización y mantenimiento. También Alcove, fundada en 1997 en Francia, basa su negocio en proporcionar servicios de consultoría, consultoría estratégica, soporte y desarrollo para *software l*ibre.

2.3. **Licencias tipo BSD**

La licencia BSD (*Berkeley Software Distribution*) tienen su origen en la universidad californiana de Berkeley, en EE.UU.

La única obligación que impone es la de dar crédito a los autores. Permite tanto la redistribución binaria y la de los códigos fuentes, así como la modificación del programa y su integración en otro, pero no obliga a ninguna de ellas.

Su apertura permite que a partir de un programa distribuido bajo una licencia de tipo BSD pueda crearse otro programa propietario, que se distribuyera luego con una licencia restrictiva.

Los críticos de las licencias BSD ven en esta característica un peligro, ya que no se garantiza la libertad de versiones futuras de los programas. Los partidarios de la misma, por contra, ven en ella la máxima expresión de la libertad.

2.4. **Ventajas y desventajas entre las licencias GPL y BDS**

La licencia GPL está pensada para asegurar la libertad del código en todo momento, ya que un programa publicado y licenciado bajo sus condiciones nunca podrá ser propietario. Es más, ni ese programa ni modificaciones al mismo pueden ser publicadas con una licencia diferente a la propia GPL.

Los partidarios de las licencias tipo BSD ven en esta imposición un recorte de la libertad, mientras que los seguidores de la licencia GPL ven en ello una forma de asegurarse que ese *software* siempre va a ser libre.

Se puede considerar que la licencia GPL maximiza las libertades de los usuarios, mientras que las tipo BSD lo hacen para los desarrolladores.

La explotación de un programa bajo licencia libre no es excluyente, en el sentido de que puede combinarse de distintas maneras con una explotación bajo un modelo propietario. Ejemplos de estas combinaciones son:

1. Distribución propietaria durante un tiempo, luego libre. Cada nueva versión del producto se vende como *software* propietario. Pasado un tiempo (normalmente, cuando se empieza a comercializar una nueva versión), esa versión pasa a distribuirse con una licencia libre. De esta manera la empresa productora obtiene ingresos de los clientes interesados en disponer lo antes posible de nuevas versiones, y a la vez minimiza la competencia, ya que cualquier empresa que quiera competir usando ese producto sólo podrá hacerlo con la versión libre.

2. Distribución limitada durante un tiempo. En este caso, el *software* es libre desde que se comienza a distribuir. Pero como nada en una licencia libre obliga a distribuir el programa a todo el que lo quiera, lo que hace el productor es distribuirlo durante un tiempo sólo a sus clientes, que le pagan por ello (normalmente en forma de contrato de mantenimiento). Al cabo de un tiempo, el productor lo distribuye a cualquiera, por ejemplo poniéndolo en un archivo de acceso público. De esta manera, el productor obtiene ingresos de sus clientes, que perciben esta disposición preferente del *software* como un valor añadido. Naturalmente, el modelo sólo funciona si los clientes a su vez no hacen público el programa cuando lo reciben.

2.5. **Licencias especiales (propietarias) en paralelo a la distribución libre**

En estos modelos, la empresa produce un producto que distribuye al tiempo bajo licencias propietarias y libres. Las licencias propietarias le

permiten vender el producto de una forma más o menos tradicional, lo que complementa con la oferta de consultoría y desarrollos relacionados con el producto. Por ejemplo, una empresa puede distribuir un producto como *software* libre bajo la GPL, pero ofrecer también una versión propietaria (simultáneamente, y sin retraso para ninguna de las dos) para quien no quiera las condiciones de la GPL, por ejemplo, porque quiere integrar el producto con uno propietario (algo que la GPL no permite).

De cualquier modo, podemos ver que la elección de licencia para un programa libre no es una tarea fácil, que hay que tener multitud de factores en cuenta, y que de nuestra elección dependerá también la relación del programa con los programas propietarios.

Sea cual sea la elección que tomemos, en favor del *software* libre o del propietario, o de una combinación de ambos, el Derecho de Autor velará por el respeto de nuestros derechos morales y patrimoniales, protegiendo nuestra elección tal y como quede plasmada en la licencia subscrita.

CONCLUSIONES

1. *Software libre* se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*. De modo más preciso, se refiere a cuatro libertades de los usuarios del *software*:
- ejecutar el programa para cualquier propósito;
- estudiar el funcionamiento del programa para adaptarlo a cualquier necesidad;
- redistribuir copias; y
- mejorar el programa y poner las mejoras a disposición del público.

2. Stallman publica el Manifiesto GNU, que es una suerte de acta fundacional o declaración de principios tanto del proyecto GNU como de la FSF. El objetivo del proyecto GNU era hacer ese sistema operativo de modo que nadie tuviera que pagar por el *software* y generar una comunidad a partir de él. Stallman lanzó el proyecto GNU porque

esencialmente pensaba que el conocimiento que constituye un programa corriente –que la informática llama código fuente– debe ser libre.

3. Se propuso *open source software* como nueva expresión para designar al *software* libre, porque además de evitar la ambigüedad del término inglés *free* (libre y gratis), permitía diferenciarlo de las ideas y de la comunidad de Stallman.

4. Dentro de *software* libre hay, a su vez, matices que es necesario tener en cuenta. Por ejemplo, el *software* de dominio público significa que no está protegido por el *copyright*, por lo tanto, podrían generarse versiones no libres del mismo, en cambio el *software* libre protegido con *copyleft* impide a los redistribuidores incluir algún tipo de restricción a las libertades propias del *software* así concebido, es decir, garantiza que las modificaciones seguirán siendo *software* libre. También es conveniente no confundir el *software* libre con el *software* gratuito, este último no cuesta nada, hecho que no lo convierte en *software* libre, porque no es una cuestión de precio, sino de libertades.

5. A diferencia del dominio público, el *copyleft* no implica "no protegido con derecho de autor", "sin derechos de autor" o "fuera del alcance de las leyes de derecho de autor": Al contrario, bajo este régimen jurídico una persona determinada tiene la titularidad de los derechos de autor y goza, en consecuencia, de un derecho exclusivo a la utilización del *software*. Y en este sentido, el *copyleft* no se diferencia en nada del sistema de *copyright o derechos de autor*. Pero, lo que otorga su especificidad al *copyleft* es la manera en que ese titular determinado ejerce los derechos que la ley le reconoce.

6. Los partidarios de las licencias tipo BSD ven en esta imposición un recorte de la libertad, mientras que los seguidores de la licencia GPL ven en ello una forma de asegurarse que ese *software* siempre va a ser libre. Se puede considerar que la licencia GPL maximiza las libertades de los usuarios, mientras que las tipo BSD lo hacen para los desarrolladores.

Referencias

ARGUDO, E. (2004). Los derechos de propiedad intelectual y la protección del software. El software libre: significado, desafíos y oportunidades.¿Es una alternativa viable como herramienta de apoyo a los desarrollos informáticos de las Oficinas de Propiedad Intelectual? En: Reunión Regional de Directores de Oficinas de Propiedad Industrial y de Oficinas de Derechos de Autor de América Latina, Guadalajara, México, 23 al 25 de marzo de 2004, (OMPI; IMPI; INDAUTOR de México), 1-7 p.

CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN (2008). Origen del *Software Libre*. Disponible en: www.cnti.gob.ve (Consultado: 06-01-08).

STALLMAN, R. (a) (1998). Manifiesto GNU, En: *Open sources. Voices from the open source revolution.* Disponible en: www.oreilly.com/catalog (Consultado: 11-12- 07).

RAYMOND, E. The New Hacker´s Dictionary. Glossary. Disponible en: www.catb.org (Consultado: 06-01-08).

RAYMOND, E. (1997). La catedral y el bazar, versión en castellano. Disponible en: www.sindominio.net (Consultado: 12-11-07).

CARRANZA, M. (2004). *Problemática jurídica del software libre*. Buenos Aires (Argentina): LexisNexis Argentina, pp. 23-120, 130-143.

www.wipo.int (Consultado 11-12-07).

CONVENIO DE BERNA para la protección de las obras literarias y artísticas de fecha 9 de septiembre de 1886, con reformas en 1896 (Acta de París); en 1908 (revisión de Berlín); en 1914 (Protocolo Adicional de Berna); en 1928 (revisión de Roma); en 1948 (revisión de Bruselas); en 1967 (revisión de Estocolmo) y en 1971 (revisión de París).

LIPSZYC, D. (1993). *Derechos de autor y derechos conexos*. París,Bogotá, Buenos Aires, Argentina: Ediciones UNESCO-CERLALC-ZAVALIA.

WEBBINK, M. (2003). Understanding Open Source Software. Disponible en: www.groklaw.net (Consultado: 11-12-07).

STALLMAN, R. (2004b). *Software libre para una sociedad libre*. Madrid: Editorial Traficantes de Sueños.