3. Aspectos legales

"The licenses for most software are designed to take away your freedom to share and change it."

"Las licencias de la mayoría de los programas están diseñadas para quitarte la libertad de compartirlos y cambiarlos."

GNU General Public License, versión 2

En este capítulo se presentan los principales aspectos legales relacionados con el software libre. Para ponerlos en contexto, se empieza por una pequeña introducción a los conceptos más básicos de la propiedad intelectual e industrial, antes de exponer la definición detallada de *software libre*, *software de fuente abierta* y otros conceptos relacionados. Se analizan también con cierto detalle las licencias de software libre más habituales y su impacto sobre los modelos de negocio (tema que se tratará con más detalle en el capítulo 5) y los modelos de desarrollo.

3.1. Breve introducción a la propiedad intelectual

El término *propiedad intelectual* tiene varias acepciones según el contexto y quién lo utiliza. Hoy en día se usa en muchos foros para agrupar distintos privilegios que se otorgan sobre bienes intangibles con valor económico. Entre ellos podemos destacar los de *copyrigh*t (derechos de autor) y similares, que protegen de la copia no autorizada trabajos literarios o artísticos, programas de ordenador, recopilaciones de datos, diseños industriales, etc.; las marcas, que protegen símbolos; las indicaciones geográficas, que protegen denominaciones de origen; los secretos industriales, que respaldan la ocultación de información, y las patentes, que otorgan monopolios temporales sobre invenciones a cambio de desvelarlas. Sin embargo, en muchas tradiciones legales, entre ellas la hispana, se distingue entre la *propiedad intelectual*, que se refiere exclusivamente a los derechos de autor, y la *propiedad industrial,* que abarca las figuras restantes.

En cualquier caso, la legislación que se aplica en todos estos aspectos es una de las más coordinadas en prácticamente todo el mundo. Por un lado, la OMPI (Organización Mundial de la Propiedad Intelectual, WIPO según sus siglas en inglés) promueve ambos tipos de propiedad en todos sus aspectos. Por otro, el acuerdo TRIPS (aspectos comerciales de la propiedad intelectual) establece unos mínimos de protección y obliga a todos los países miembros de la OMC (Organización Mundial del Comercio, WTO) a desarrollarlos en unos ciertos plazos, que dependen del nivel de desarrollo del país[3] .

(3)El acuerdo TRIPS fue firmado por la presión de los países industrializados (especialmente Estados Unidos y Japón).

La Declaración Universal de los Derechos Humanos reconoce, en su artículo 27, el derecho a que se protejan los intereses morales y materiales que correspondan a cualquier persona por razón de las producciones científicas, literarias o artísticas de que sean autores. Sin embargo, en muchos casos (y de forma habitual en el caso del software), este derecho suele ser transferido en la práctica a las empresas que emplean a los creadores o que comercializan sus creaciones. No obstante, la propiedad intelectual se justifica no sólo por razones morales, sino también por razones prácticas, para dar cumplimiento a otro derecho: el de la sociedad a beneficiarse de las creaciones, incentivándolas con beneficios y protegiendo las inversiones para la creación, la investigación y el desarrollo. Para armonizar ambos derechos, la propiedad intelectual es temporal, y caduca cuando ha cumplido su función de promoción.

Pero la caducidad no es la única característica que diferencia la propiedad intelectual de la ordinaria. Hoy en día, los objetos de la misma pueden copiarse fácil y económicamente, sin pérdida de calidad. La copia no perjudica a quien ya disfruta de lo copiado, al contrario del robo, que sí que priva del objeto al poseedor original. La copia sí que puede perjudicar al propietario, ya que lo priva potencialmente de los ingresos de una venta. El control de la copia de intangibles es mucho más complicado que el del robo de bienes tangibles y puede llevarnos a una sociedad policial, que necesite controlar todas las copias de información, y a una gran inseguridad jurídica, porque aumentan las posibilidades de violación accidental de derechos. Además la creatividad es incremental: al crear siempre se copia algo, y la línea divisoria entre la copia burda y la inspiración es sutil.

Para profundizar más en todo esto, en los siguientes apartados se repasan algunas de las categorías de la propiedad intelectual. En cualquier caso, se puede adelantar ya que el software libre propone un nuevo punto de equilibrio en este ámbito, primando los beneficios de la copia y la innovación incremental frente al control exclusivo de una obra por parte de su autor.

3.1.1. Derechos de autor

Los derechos de autor (*copyright*) protegen la expresión de un contenido, no el contenido en sí mismo. Se desarrollaron para recompensar a los autores de libros o de arte. Las obras protegidas pueden expresar ideas, conocimientos o métodos libremente utilizables, pero se prohíbe reproducirlas sin permiso, total o parcialmente, con o sin modificaciones. Esta protección es muy sencilla, ya que entra automáticamente en vigor con ámbito casi universal en el momento de publicación de la obra. Modernamente se ha extendido a los programas de ordenador y (en algunas áreas geográficas) a recopilaciones de datos.

La Ley de Propiedad Intelectual (LPI) en España, y leyes similares en otros países, desarrolladas sobre la base del Convenio de Berna para la protección de trabajos literarios y artísticos de 1886, regulan los derechos de autor. Estos derechos se dividen en derechos morales y derechos patrimoniales. Los primeros

garantizan al autor el control sobre la divulgación de su obra, con nombre o seudónimo, el reconocimiento de autoría, el respeto a la integridad de la obra y el derecho de modificación y retirada. Los segundos le dan derecho a explotarla económicamente y pueden ser cedidos total o parcialmente, de forma exclusiva o no, a un tercero. Los derechos morales son vitalicios o indefinidos, mientras que los patrimoniales tienen una duración bastante larga (setenta años después de la muerte del autor, si es una persona física, en el caso de la ley española).

La cesión de derechos se especifica mediante un contrato denominado *licencia*. En el caso de programas privativos, éstos generalmente se distribuyen por medio de licencias de uso "no exclusivo", que se entiende que se aceptan automáticamente al abrir o instalar el producto. No es necesario pues firmar el contrato, ya que, en el caso de no aceptarlo el receptor, rigen automáticamente los derechos por omisión de la ley, es decir, ninguno. Las licencias no pueden restringir algunos derechos que otorga la legislación vigente, como el de hacer copias privadas de arte o música, lo que permite regalar una copia de una grabación a un amigo, pero este derecho no es aplicable a los programas. Según la LPI de 1996 (Ley de Propiedad Intelectual. Real Decreto Legislativo 1/1996, de 12 de abril) [77], modificada en 2006 (Ley de Propiedad Intelectual. Ley 23/2006, de 7 de julio) [79], respecto de los programas siempre se puede hacer una copia de seguridad, se pueden estudiar para hacer programas interoperables y se pueden corregir y adaptar a nuestras necesidades (cosa difícil, porque no solemos disponer de los códigos fuente). Estos derechos no pueden ser restringidos por licencias, aunque las leyes están en proceso de revisión, en una tendencia aparentemente imparable a reducir los derechos de los usuarios. Las recopilaciones organizadas de obras o datos ajenos también están sometidas a derechos de autor, si bien los términos son distintos y la duración menor.

Las nuevas tecnologías de la información, y en especial la Red, han trastocado profundamente la protección de los derechos de autor, ya que las expresiones de contenidos son mucho más fáciles de copiar que los contenidos mismos. Y en el caso de los programas y algunas obras de arte (música, imágenes, películas, e incluso literatura), "funcionan" automáticamente en el ordenador, sin necesidad de un esfuerzo humano apreciable. En cambio, los diseños o inventos hay que construirlos, y posiblemente ponerlos en producción. Esta posibilidad de crear riqueza sin coste ha llevado a gran parte de la sociedad, en particular a los países pobres, a duplicar programas sin pagar licencia, sin que exista una conciencia social de que eso sea una "mala acción" (como sí que la suele haber con respecto al robo de bienes físicos, por ejemplo). Por otro lado, los fabricantes de programas, solos o en coalición (por ejemplo la BSA, Business Software Alliance), presionan fuertemente para que las licencias se paguen y los gobiernos persigan lo que se ha dado en llamar *piratería*.

Nota

La palabra *piratería* se ha popularizado como sinónimo de 'violación de cualquier forma de propiedad intelectual, especialmente en el caso de la copia ilegal de programas, música y películas'. El término parece exagerado, y en el diccionario de la Real Academia Española de la Lengua aparece como una acepción en sentido figurado, ya que el término original se refiere a 'robo con violencia en el mar'. Por ello Richard Stallman recomienda evitarlo ("Some confusing or loaded words and phrases that are worth avoiding", 2003) [212].

Precisamente para proteger los derechos de autor de aquellos contenidos con licencias privativas, nacen los llamados sistemas DRM (*digital rights management*, 'gestión de derechos digitales'), con el fin de controlar el acceso y la utilización de datos en soporte digital o de restringir su uso a ciertos dispositivos. El empleo de sistemas DRM se ha criticado fuertemente en muchos sectores, puesto que trata de proteger derechos de autor imponiendo restricciones más allá de las suficientes, por lo que algunos, como la Free Software Foundation, recomiendan interpretar las siglas como *digital restrictions management* ('gestión de restricciones digitales'), intentando evitar la utilización de la palabra *derechos* (en inglés, *rights*), al considerar que se privan excesivos derechos de los usuarios para lograr satisfacer los derechos de los autores.

3.1.2. Secreto comercial

Otro de los recursos que tienen las empresas para rentabilizar sus inversiones es el secreto comercial, protegido por las leyes de propiedad industrial, siempre que las empresas tomen las medidas suficientes para ocultar la información que no quieren desvelar. En el caso de productos químicos o farmacéuticos que requieran aprobación gubernamental, el Estado se compromete a no desvelar los datos entregados que no sea obligatorio hacer públicos.

Una de las aplicaciones más conocidas del secreto comercial se encuentra en la industria del software propietario, que generalmente comercializa programas compilados sin dar acceso al código fuente, para así impedir el desarrollo de programas derivados.

A primera vista parece que la protección del secreto comercial es perversa, ya que puede privar indefinidamente a la sociedad de conocimientos útiles. En cierto modo así lo entienden algunas legislaciones, permitiendo la ingeniería inversa para desarrollar productos sustitutos, aunque la presión de las industrias ha conseguido que en muchos países ésta sea una actividad prohibida y en otros sólo esté permitida en aras de la compatibilidad.

Sea perverso o no el secreto comercial, en muchos casos es mejor que una patente, ya que da una ventaja competitiva al que pone un producto en el mercado mientras la competencia trata de imitarlo con ingeniería inversa. Cuanto más sofisticado sea el producto, más costará a la competencia reproducirlo, mientras que si es trivial, lo copiará rápidamente. La imitación con mejoras

ha sido fundamental para el desarrollo de las que hoy son superpotencias (Estados Unidos y Japón) y es muy importante para la independencia económica de los países en vías de desarrollo.

3.1.3. Patentes y modelos de utilidad

La alternativa al secreto comercial es la patente. A cambio de un monopolio de diecisiete a veinticinco años y un determinado coste económico, un *invento* es revelado públicamente, de forma que sea fácilmente reproducible. Con ella se pretende promover la investigación privada, sin coste para el contribuyente y sin que el resultado se pierda. El poseedor de una patente puede decidir si permite a otros utilizarla y el precio que debe pagar por la licencia.

La doctrina oficial es que el sistema de patentes promueve la innovación, pero cada vez más se hacen oír voces que afirman que la dificulta, bien porque opinan que el sistema está mal implementado, o bien porque creen que es perverso en sí mismo (François-René Rideau, "Patents are an economic absurdity", 2000) [194].

Lo que se considera un invento ha ido variando con el tiempo, y existen grandes presiones para ampliar la cobertura del sistema, que incluyen algoritmos, programas, modelos de negocio, sustancias naturales, genes y formas de vida, incluidas plantas y animales. TRIPS exige que el sistema de patentes no discrimine ningún ámbito del saber. Las presiones de la Organización Mundial de la Propiedad Intelectual (OMPI o WIPO) pretenden eliminar la necesidad de que el invento tenga aplicación industrial, y también rebajar los estándares de inventiva exigibles en una patente. Estados Unidos está a la cabeza de los países con un mínimo de exigencias sobre lo que es patentable, y es además el más beligerante para que otros países adopten sus estándares, sin acordarse de que él mismo se negó a aceptar las patentes extranjeras cuando era un país subdesarrollado.

Una vez obtenida una patente, los derechos del poseedor son independientes de la calidad del invento y del esfuerzo invertido en obtenerlo. Dado el coste de mantenimiento de una patente y los costes de litigación, solamente las grandes empresas pueden mantener y mantienen una amplia cartera de patentes que las sitúan en una posición que les permite ahogar cualquier competencia. Dada la facilidad para colocar patentes sobre soluciones triviales o de gran aplicabilidad, pueden monopolizar para sí un espacio muy amplio de actividad económica.

Con patentes, muchas actividades, especialmente la programación, se hacen extremadamente arriesgadas, ya que es muy fácil que en el desarrollo de un programa complicado se viole accidentalmente alguna patente. Cuando dos o más empresas están investigando para resolver un problema, es muy probable que lleguen a una solución similar casi al mismo tiempo, pero sólo una (generalmente la que tenga más recursos) logrará patentar su invento, de manera

que las otras perderán toda posibilidad de rentabilizar su inversión. Todo desarrollo técnico complejo puede convertirse en una pesadilla si para cada una de las soluciones de sus partes es necesario investigar si la solución encontrada está patentada (o en trámite), para intentar obtener la licencia o para buscar una solución alternativa. Este problema es especialmente grave en el software libre, donde las violaciones de patentes de algoritmos son evidentes por simple inspección del código.

Aunque en Europa aún es ilegal patentar un algoritmo, se podrá hacer en muy breve plazo, quizá cuando el lector lea estas líneas.

3.1.4. Marcas y logotipos registrados

Las marcas y logotipos son nombres y símbolos que representan un acervo de calidad (o una gran inversión en publicidad). No tienen gran importancia dentro del software libre, posiblemente porque registrarlos tiene un coste. Así, solamente algunos nombres importantes, como Open Source (por Open Source Foundation), Debian (por Software in the Public Interest), GNOME (por GNOME Foundation), GNU (por Free Software Foundation) u OpenOffice.org (por SUN Microsystems) están registrados, y sólo en algunos países. Sin embargo, el no registro de nombres ha provocado problemas. Por ejemplo, en Estados Unidos (1996) y en Corea (1997) ha habido personas que han registrado el nombre Linux y han demandado dinero por su uso. La resolución de estas disputas supone costes legales y la necesidad de demostrar un uso del nombre anterior a la fecha del registro.

3.2. Licencias en el software libre

Legalmente hablando, la situación de los programas libres respecto de los privativos no es muy diferente: también se distribuyen bajo licencia. Lo que los diferencia es precisamente lo que permite esa licencia. En el caso de las licencias de programas libres, que no restringen precisamente el uso, la redistribución y la modificación, lo que pueden imponer son condiciones que hay que satisfacer precisamente en caso de que se quiera redistribuir el programa. Por ejemplo, pueden exigir que se respeten las indicaciones de autoría o que se incluya el código fuente si se quiere redistribuir el programa listo para ejecutar.

Aunque en esencia *software libre* y *software propietario* se diferencien en la licencia con la que los autores publican sus programas, es importante hacer hincapié en que esta diferencia se refleja en condiciones de uso y redistribución totalmente diferentes. Como se ha visto a lo largo de los últimos años, esto ha originado no sólo métodos de desarrollo totalmente diferentes, sino incluso formas prácticamente opuestas (en muchos sentidos) de entender la informática.

Las leyes sobre propiedad intelectual aseguran que en ausencia de permiso explícito no se puede hacer casi nada con una obra (en nuestro caso, un programa) que se reciba o se compre. Sólo el autor (o el que posea los derechos de la obra) nos puede dar ese permiso. En cualquier caso, la propiedad de la obra no cambia por otorgar una licencia, ya que ésta no supone transferencia de propiedad, sino solamente de derecho de uso y, en algunos casos (obligados en el software libre), de distribución y modificación. Las licencias de software libre se diferencian de las privativas precisamente en que en lugar de restringir cuidadosamente lo que se permite, otorgan ciertos permisos explícitos. Cuando uno recibe un programa libre puede redistribuirlo o no, pero si lo redistribuye, sólo puede hacerlo porque la licencia se lo permite. Pero para ello es preciso cumplir con la licencia. En definitiva, la licencia contiene las normas de uso a las que han de atenerse usuarios, distribuidores, integradores y otras partes implicadas en el mundo de la informática.

Para comprender plenamente todos los entresijos legales que se van a presentar en este capítulo (y que, sin duda, son muy importantes para entender la naturaleza del software libre), también es necesario saber que cada nueva versión de un programa es considerada como una nueva obra. El autor tiene, otra vez, plena potestad para hacer con ella lo que le apetezca, incluso distribuirla en términos y condiciones totalmente diferentes (o sea, con una licencia diferente a la anterior). Así, si el lector es autor único de un programa podrá publicar una versión bajo una licencia de software libre y, si le apeteciese, otra posterior bajo una licencia propietaria. En caso de que existan más autores y que la nueva versión contenga código cuya autoría les corresponda, si se va a publicar bajo otras condiciones, todos ellos habrán de dar el visto bueno al cambio de licencia.

Un tema todavía relativamente abierto es la licencia que se aplica a las contribuciones externas. Generalmente se supone que una persona que contribuya al proyecto acepta *de facto* que su contribución se ajuste a las condiciones especificadas por su licencia, aunque esto podría tener poco fundamento jurídico. La iniciativa de la Free Software Foundation de pedir mediante carta (física) la cesión de todos los derechos de *copyright* a cualquiera que contribuya con más de diez líneas de código a un subproyecto de GNU es una buena muestra de que en el mundo del software libre hay políticas más estrictas con respecto a estas contribuciones.

Partiendo de todo lo dicho, vamos a centrarnos ya en el resto de este capítulo en el análisis de diversas licencias. Para poner en contexto este estudio, hay que recordar que de ahora en adelante, cuando decimos que una licencia es de software libre, lo decimos en el sentido de que cumple las definiciones de *software libre* presentadas en el apartado 1.1.1.

3.2.1. Tipos de licencias

La variedad de licencias libres es grande, aunque por razones prácticas la mayoría de los proyectos utilizan un pequeño conjunto de cuatro o cinco. Por un lado, muchos proyectos no quieren o no pueden dedicar recursos a diseñar una licencia propia; por otro, la mayoría de los usuarios prefieren referirse a una licencia ampliamente conocida que leerse y analizar licencias completas.

Bibliografía

Se pueden ver recopiladas y comentadas tanto licencias consideradas libres como licencias consideradas no libres o libres pero incompatibles con la GPL desde el punto de vista de la FSF en Free Software Foundation, "Licencias libres" [121]. El punto de vista filosóficamente diferente de la Open Source Initiative se refleja en su listado (Open Source Initiative, "Licencias de fuente abierta") [181]. Pueden verse discrepancias en algunas licencias, como la Apple Public Source License Ver. 1.2, considerada no libre por la FSF por la obligación de publicar todos los cambios (aunque sean privados), de notificar a Apple las redistribuciones, o por la posibilidad de revocación unilateral. No obstante, la presión de esta clasificación hizo que Apple publicara la versión 2.0 en agosto de 2003, ya considerada libre por la FSF.

Es posible dividir las licencias de software libre en dos grandes familias. La primera está compuesta por las licencias que no imponen condiciones especiales en la *segunda redistribución* (esto es, que sólo especifican que el software se puede redistribuir o modificar, pero que no imponen condiciones especiales si se hace, lo que permite, por ejemplo, que alguien que reciba el programa pueda después redistribuirlo como software propietario): son las que llamaremos *licencias permisivas*. La segunda familia, que denominaremos *licencias robustas* (o *licencias* copyleft), incluye las que, al estilo de la GNU GPL, imponen condiciones en caso de que se quiera redistribuir el software, condiciones que van en la línea de forzar a que se sigan cumpliendo las condiciones de la licencia después de la *primera redistribución*. Mientras que el primer grupo hace énfasis en la libertad de quien recibe un programa, que le permite hacer casi todo lo que quiera con él (en términos de condiciones de futuras redistribuciones), el segundo hace énfasis en la libertad de cualquiera que potencialmente pueda recibir algún día un trabajo derivado del programa, que obliga a que las sucesivas modificaciones y redistribuciones respeten los términos de la licencia original.

Nota

El término *copyleft* aplicado a una licencia, usado sobre todo por la Free Software Foundation para definir las suyas, tiene implicaciones similares a las de la expresión *licencia robusta* tal como la usamos en este texto.

La diferencia entre estos dos tipos de licencias ha sido (y es) tema de debate en la comunidad del software libre. En cualquier caso, es conveniente recordar que todas ellas son licencias libres.

3.2.2. Licencias permisivas

Las licencias permisivas, a veces también llamadas *licencias liberales* o *minimalistas*, no imponen prácticamente ninguna condición sobre quien recibe el software, y sin embargo, le dan permiso de uso, redistribución y modificación. Desde cierto punto de vista, este enfoque puede entenderse como la garantía de las máximas libertades para quien recibe un programa. Pero desde otro, puede entenderse también como la máxima despreocupación con respecto al

hecho de que una vez recibido un programa por parte de alguien, se sigan garantizando las mismas libertades cuando ese programa se redistribuya. De hecho, típicamente estas licencias permiten que se redistribuya con licencia privativa un software cuyo autor distribuye con licencia permisiva.

Entre estas licencias, una de las más conocidas es la licencia BSD, hasta tal punto que muchas veces se llama a las licencias permisivas *licencias de tipo BSD*. La licencia BSD (Berkeley Software Distribution) tiene su origen en la publicación de versiones de Unix realizadas por la universidad californiana de Berkeley, en EE.UU. La única obligación que exige es dar crédito a los autores, mientras que permite tanto la redistribución binaria como la de los códigos fuente, aunque no obliga a ninguna de las dos en ningún caso. Asimismo, da permiso para realizar modificaciones y ser integrada con otros programas casi sin restricciones.

Nota

Una de las consecuencias prácticas de las licencias de tipo BSD ha sido la difusión de estándares, ya que los desarrolladores no encuentran ningún obstáculo para realizar programas compatibles con una implementación de referencia bajo este tipo de licencias. De hecho, ésta es una de las razones de la extraordinaria y rápida difusión de los protocolos de Internet y de la interfaz de programación basada en zócalos (*sockets*), porque la mayoría de los desarrolladores comerciales derivó su realización de la de la Universidad de Berkeley.

Las licencias permisivas son bastante populares, y existe toda una familia con características similares a la BSD: X Window, Tcl/Tk, Apache, etc. Históricamente estas licencias aparecieron debido a que el software correspondiente fue creado en universidades con proyectos de investigación financiados por el Gobierno de los Estados Unidos. Estas universidades prescindían de la comercialización de estos programas, asumiendo que ya habían sido pagados previamente por el Gobierno, y por tanto, con los impuestos de todos los contribuyentes, por lo que cualquier empresa o particular podía utilizar el software casi sin restricciones.

Como ya se ha comentado, a partir de un programa distribuido bajo una licencia permisiva puede crearse otro (en realidad, una nueva versión) que sea privativo. 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. Sus partidarios, por el contrario, la consideran la máxima expresión de la libertad y argumentan que, a fin de cuentas, se puede hacer (casi) todo lo que se quiera con el software.

La mayoría de las licencias permisivas son una copia calcada de la original de Berkeley en la que se modifica todo lo referente a la autoría. En algunos casos, como la licencia del proyecto Apache, incluyen alguna cláusula adicional, como la imposibilidad de llamar a las versiones redistribuidas de igual manera

que a la original. Todas estas licencias suelen incluir, como la BSD, la prohibición de usar el nombre del propietario de los derechos para promocionar productos derivados.

Asimismo, todas las licencias, sean de tipo BSD o no, incluyen una *limitación de garantía* que es en realidad una *negación de garantía*, necesaria para evitar demandas legales por garantías implícitas. Aunque se ha criticado mucho esta negación de garantía en el software libre, es práctica habitual en el software propietario, que generalmente sólo garantiza que el soporte es correcto y que el programa en cuestión se ejecuta.

Esquema resumen de la licencia BSD

Copyright © *el propietario*. Todos los derechos reservados.

Se permite la redistribución en fuente y en binario, con o sin modificación, siempre que se cumplan las condiciones siguientes:

1) Las redistribuciones en fuente deben retener la nota de *copyright* y listar estas condiciones y la limitación de garantía.

2) Las redistribuciones en binario deben reproducir la nota de *copyright* y listar estas condiciones y la limitación de garantía en la documentación.

3) Ni el nombre del *propietario* ni el de los que han contribuido pueden usarse sin permiso para promocionar productos derivados de este programa.

Este programa se proporciona "tal cual", sin garantías expresas ni implícitas, tales como su aplicabilidad comercial o su adecuación para un propósito determinado. En ningún caso *el propietario* será responsable de ningún daño causado por su uso (incluida la pérdida de datos, la pérdida de beneficios o la interrupción de negocio).

A continuación describimos brevemente algunas licencias permisivas:

- Licencia de X Window, versión 11 (X11)
  (http://www.x.org/Downloads_terms.html) [73].
  Es la licencia usada para la distribución del sistema X Window, el sistema de ventanas más ampliamente utilizado en el mundo Unix, y también en entornos GNU/Linux. Es muy similar a la licencia BSD, que permite redistribución, uso y modificación prácticamente sin restricciones. A veces se la llama *licencia MIT* (con peligrosa poca precisión, porque el MIT ha usado otros tipos de licencias). Bajo esta licencia se distribuyen también trabajos derivados de X Windows, como XFree86.

- Zope Public License 2.0 (http://www.zope.org/Resources/ZPL) [76].
  Esta licencia (habitualmente llamada ZPL) se usa para la distribución de Zope (un servidor de aplicaciones) y otros productos relacionados. Es similar a la BSD, con el interesante detalle de que prohíbe expresamente el uso de marcas registradas por Zope Corporation.

- Licencia de Apache.
  Es una licencia bajo la que se distribuyen la mayor parte de los programas producidos por el proyecto Apache. Es similar a la licencia BSD.

Hay algunos programas libres que no se distribuyen con una licencia específica, sino que su autor los declara explícitamente *public domain* ('de dominio público o del común'). La principal consecuencia de esta declaración es que el autor renuncia a todos sus derechos sobre el programa, que por lo tanto, se puede modificar, redistribuir, usar, etc. de cualquier manera. A efectos prácticos, es muy similar a que el programa esté bajo una licencia de tipo BSD.

3.2.3. Licencias robustas

La Licencia Pública General de GNU (GNU GPL)

La Licencia Pública General del proyecto GNU (Free Software Foundation, 1991) [118] (más conocida por su acrónimo en inglés, GPL), que mostramos traducida en el apéndice C, es con diferencia la más popular y conocida de todas las del mundo del software libre. Su autoría corresponde a la Free Software Foundation (promotora del proyecto GNU), y en un principio fue creada para ser la licencia de todo el software generado por la FSF. Sin embargo, su utilización ha ido más allá hasta convertirse en la licencia más utilizada (por ejemplo, más del 70% de los proyectos anunciados en Freshmeat están licenciados bajo la GPL), incluso por proyectos bandera del mundo del software libre, como el núcleo Linux.

La licencia GPL es interesante desde el punto de vista legal porque hace un uso muy creativo de la legislación de *copyright*, consiguiendo efectos prácticamente contrarios a los que se suponen de la aplicación de esta legislación: en lugar de limitar los derechos de los usuarios, los garantiza. Por este motivo, en muchos casos se denomina a esta "maniobra" *copyleft* (juego de palabras en inglés que se puede traducir como 'izquierdos de autor'). Alguien con una pizca de humor llegó incluso a lanzar el eslogan "copyleft, all rights reversed".

En líneas básicas, la licencia GPL permite la redistribución binaria y la del código fuente, aunque en el caso de que redistribuya de manera binaria obliga a que también se pueda acceder a los códigos fuente. Asimismo, está permitido realizar modificaciones sin restricciones. Sin embargo, sólo se puede redistribuir código licenciado bajo GPL de forma integrada con otro código (por ejemplo, mediante enlaces o *links*) si éste tiene una licencia compatible. Esto se ha llamado *efecto viral* (aunque muchos consideran despectiva esta denominación) de la GPL, ya que un código publicado una vez con esas condiciones nunca puede cambiarlas.

Nota

Una licencia es incompatible con la GPL cuando restringe alguno de los derechos que la GPL garantiza, bien explícitamente, contradiciendo alguna cláusula, o bien implícitamente, imponiendo alguna nueva. Por ejemplo, la licencia BSD actual es compatible, pero la de Apache, que exige que se mencione explícitamente en los materiales de publicidad que el trabajo combinado contiene código de todos y cada uno de los titulares de derechos, es incompatible. Esto no implica que no se puedan usar simultáneamente programas con ambas licencias, o incluso integrarlos. Sólo supone que esos programas integrados no se pueden distribuir, pues es imposible cumplir simultáneamente las condiciones de redistribución de ambos.

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 privativo. Es más, ni ese programa ni modificaciones del mismo pueden ser publicados con una licencia diferente de la propia GPL. Como ya se ha dicho, los partidarios de las licencias de tipo BSD ven en esta cláusula un recorte de la libertad, mientras que sus seguidores creen que es una forma de asegurarse que ese software siempre va a ser libre. Por otro lado, se puede considerar que la licencia GPL maximiza las libertades de los usuarios, mientras que las de tipo BSD maximizan las de los desarrolladores. Hay que observar, sin embargo, que en el segundo caso estamos hablando de los desarrolladores en general y no de los autores, ya que muchos autores consideran que la licencia GPL es más beneficiosa para sus intereses, puesto que obliga a sus competidores a publicar sus modificaciones (mejoras, correcciones, etc.) en caso de que redistribuyan su software, mientras que con una licencia de tipo BSD éste no tiene por qué ser el caso.

En cuanto a la naturaleza contraria al *copyright* de esta licencia, se debe a que su filosofía (y la de la Free Software Foundation) es que el software no debe tener propietarios (Richard Stallman, "Why software should not have owners", 1998) [207]. Aunque es cierto que el software licenciado con la GPL tiene un autor, que es el que a fin de cuentas permite la aplicación de la legislación de *copyright* sobre este software, las condiciones bajo las que lo publica le confieren tal carácter que podemos considerar que la propiedad del software corresponde a quien lo tiene y no a quien lo ha creado.

Por supuesto, esta licencia también incluye *negaciones de garantía* para proteger a los autores. Asimismo, y para preservar la buena fama de los autores originales, toda modificación de un fichero fuente debe incluir una nota en la que se especifique la fecha y el autor de la modificación.

La GPL tiene en cuenta también las patentes de software, y exige que si el código lleva algoritmos patentados (como hemos dicho, algo legal y usual en Estados Unidos y práctica irregular en Europa), o se concede licencia de uso de la patente libre de tasas, o no se puede distribuir bajo la GPL.

La última versión de la licencia GPL, la segunda, se publicó en 1991 (aunque en el momento de escribir este texto está en avanzado proceso de preparación la tercera). Precisamente teniendo en cuenta futuras versiones, la licencia recomienda licenciar bajo las condiciones de la segunda o de cualquier otra

posterior publicada por la Free Software Foundation, cosa que hacen muchos autores. Sin embargo, otros, entre los que destaca Linus Torvalds (creador de Linux), publican su software sólo bajo las condiciones de la segunda versión de la GPL, buscando desmarcarse de las posibles evoluciones futuras de la Free Software Foundation.

La tercera versión de la GPL (http://gplv3.fsf.org) [115] pretende llevarla al escenario actual del software, principalmente en aspectos como patentes, sistemas DRM (*digital rights management*, 'gestión de derechos digitales') y otras limitaciones de la libertad del software. Por ejemplo, en el borrador disponible en el momento de escribir este texto (mayo de 2007), no permite que un fabricante de hardware bloquee la utilización de ciertos módulos de software si no presentan una firma digital que acredite una determinada autoría. Un ejemplo de estas prácticas se da en los grabadores digitales de vídeo TiVo, que proporcionan el código fuente de todo su software (licenciado con GPLv2) al tiempo que no permiten que se utilicen modificaciones del código en dicho hardware[4].

(4)Este caso ha llegado incluso a sugerir la denominación de *tivoisation* para varios otros similares que han surgido.

La licencia tampoco permite que el software obligue a la ejecución en entorno prefijado, como ocurre cuando se prohíbe la utilización de núcleos no firmados en distribuciones cuya política de seguridad lo considere oportuno.

Nota

Hay varios puntos en la licencia GPLv3 que han despertado una cierta oposición. Uno de los grupos de opositores está compuesto por desarrolladores del núcleo Linux (entre ellos el propio Linus Torvalds). Consideran que el requisito de utilización de componentes de software firmados permite otorgar ciertas características de seguridad imposibles de otra manera, al tiempo que su prohibición explícita extendería la licencia al terreno del hardware. Además, la limitación establecida por el mecanismo de firmas se daría únicamente en las plataformas de hardware y software así diseñadas, de manera que se podría modificar el software para su utilización en otro hardware. Con respecto a este punto, la FSF está a favor del empleo de mecanismos de firmas que recomienden la no utilización de componentes no firmados por motivos de seguridad, pero cree que la no prohibición de aquellos mecanismos de firmas que imposibilitan la utilización de componentes no firmados podrían dar lugar a escenarios en los que no existiesen plataformas de hardware o software en las que ejecutar dichas modificaciones del software, por lo que en ese caso quedarían totalmente limitadas las libertades del software libre en lo que a modificación del código se refiere.

La Licencia Pública General Menor de GNU (GNU LGPL)

La Licencia Pública General Menor del proyecto GNU (Free Software Foundation, GNU Lesser General Public License, versión 2.1, febrero 1999) [119], comúnmente conocida por sus iniciales en inglés, LGPL, es la otra licencia de la Free Software Foundation. Pensada al principio para su uso en bibliotecas (la L, en sus inicios, venía de *library*, 'biblioteca'), fue modificada recientemente para ser considerada la hermana menor (*lesser*, 'menor') de la GPL.

La LGPL permite el uso de programas libres con software propietario. El programa en sí se redistribuye como si estuviera bajo la licencia GPL, pero se permite su integración con cualquier otro paquete de software sin prácticamente limitaciones.

Como se puede ver, en un principio esta licencia estaba orientada a las bibliotecas, para que se pudiera potenciar su uso y desarrollo sin tener los problemas de integración que implica la GPL. Sin embargo, cuando se vio que el efecto buscado de popularizar las bibliotecas libres no se veía compensado por la generación de programas libres, la Free Software Foundation decidió el cambio de *library* a *lesser* y desaconsejó su uso, salvo para condiciones muy puntuales y especiales. Hoy en día, existen muchos programas que no son bibliotecas licenciados bajo las condiciones de la LGPL. Por ejemplo, el navegador Mozilla o el paquete ofimático (*suite*) OpenOffice.org están licenciados, entre otros, también bajo la LGPL.

Nota

Igual que pasa con la GPL, la última versión publicada de la LGPL es la segunda, aunque ya hay un borrador de la tercera versión (http://gplv3.fsf.org/pipermail/info-gplv3/2006-July/000008.html) [116]. Esta nueva versión es más corta que la anterior, dado que refiere todo su texto a la GPLv3 y destaca únicamente sus diferencias.

Otras licencias robustas

Otras licencias robustas que puede resultar interesante comentar son las siguientes:

- Licencia de Sleepycat (www.sleepycat.com/download/oslicense.html) [59].
  Es la licencia bajo la que la empresa Sleepycat (http://www.sleepycat.com/) [60] distribuye sus programas (entre los que destaca el conocido Berkeley DB). Obliga a ciertas condiciones siempre que se redistribuya el programa o trabajos derivados del mismo. En particular, obliga a ofrecer el código fuente (incluidas las modificaciones si se trata de un trabajo derivado) y a que la redistribución imponga al receptor las mismas condiciones. Aunque mucho más corta que la GNU GPL, es muy similar a ella en sus efectos principales.

- eCos License 2.0 (http://www.gnu.org/licenses/ecos-license.html) [25].
  Es la licencia bajo la que se distribuye eCos (http://sources.redhat.com/ecos/) [24], un sistema operativo en tiempo real. Es una modificación de la GNU GPL que no considera que el código que se enlace con programas protegidos por ella queden sujetos a las cláusulas de la GNU GPL si se redistribuyen. Desde este punto de vista, sus efectos son similares a los de la GNU LGPL.

- Affero General Public License (http://www.affero.org/oagpl.html) [78].

Es una interesante modificación de la GNU GPL que considera el caso de los programas que ofrecen servicios vía web, o en general, vía redes de ordenadores. Este tipo de programas plantean un problema desde el punto de vista de las licencias robustas. Como el uso del programa no implica haberlo recibido mediante una redistribución, aunque esté licenciado, por ejemplo, bajo la GNU GPL, alguien puede modificarlo y ofrecer un servicio en la Red usándolo, sin redistribuirlo de ninguna forma, y por tanto, sin estar obligado, por ejemplo, a distribuir su código fuente. La Affero GPL tiene una cláusula que obliga a que, si el programa tiene un medio para proporcionar su código fuente vía web a quien lo use, no se pueda desactivar esa característica. Esto significa que si el autor original incluye esa capacidad en el código fuente, cualquier usuario puede obtenerlo, y además esa *redistribución* está sometida a las condiciones de la licencia. La Free Software Foundation se plantea incluir provisiones similares en la versión 3 de su GNU GPL.

- IBM Public License 1.0 (http://oss.software.ibm.com/developerworks/opensource/license10.html) [40].
  Es una licencia que permite la redistribución binaria de trabajos derivados sólo si (entre otras condiciones) se prevé algún mecanismo para que quien reciba el programa pueda recibir su código fuente. La redistribución del código fuente se ha de hacer bajo la misma licencia. Además, esta licencia es interesante porque obliga al que redistribuye el programa con modificaciones a licenciar automática y gratuitamente las patentes que puedan afectar a esas modificaciones, y que sean propiedad del redistribuidor, a quien reciba el programa.

- Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html) [49].
  Se trata de un ejemplo de licencia libre con origen en una empresa. Es una evolución de la primera licencia libre que tuvo Netscape Navigator, que en su momento fue muy importante por ser la primera vez que una empresa muy conocida decidía distribuir un programa bajo su propia licencia libre.

3.2.4. Distribución bajo varias licencias

Hasta ahora se ha dado por supuesto que cada programa se distribuye bajo una única licencia en la que se especifican las condiciones de uso y redistribución. Sin embargo, un autor puede distribuir obras con distintas licencias. Para entenderlo, debemos tener en cuenta que cada publicación es una nueva obra, y que se puede dar el caso de que se distribuyan versiones que sólo difieran en la licencia. Como veremos, la mayoría de las veces esto se traduce en el hecho de que en función de lo que el usuario quiera hacer con el software se encontrará con que tiene obedecer una licencia u otra.

Uno de los ejemplos más conocidos de doble licencia es el de la biblioteca Qt, sobre la que se cimenta el entorno de escritorio KDE. Trolltech, una empresa afincada en Noruega, distribuía Qt con una licencia propietaria, aunque eximía del pago a los programas que hicieran uso de la misma sin ánimo de lucro. Por esta causa y por sus características técnicas, fue elegida a mediados de la década de los noventa por el proyecto KDE. Esto supuso una ardua polémica con la Free Software Foundation, ya que KDE dejaba de ser entonces software libre en su conjunto, al depender de una biblioteca propietaria. Tras un largo debate (durante el cual apareció GNOME como competidor libre de KDE en el escritorio), Trolltech decidió utilizar el sistema de doble licencia para su producto estrella: los programas bajo la GPL podían hacer uso de una versión de Qt GPL, mientras que si se querían integrar con programas con licencias incompatibles con la GPL (por ejemplo, licencias privativas), debían comprarles una licencia especial. Esta solución satisfizo a todas las partes, y hoy en día KDE se considera software libre.

Otros ejemplos conocidos de licencia dual son StarOffice y OpenOffice.org, o Netscape Communicator y Mozilla. En ambos casos el primer producto es propietario, mientras que el segundo es una versión libre (generalmente bajo las condiciones de varias licencias libres). Aunque en un principio los proyectos libres eran versiones limitadas de sus hermanos propietarios, con el tiempo han ido tomando su propio camino, por lo que a día de hoy tienen un grado de independencia bastante grande.

3.2.5. Documentación de programas

La documentación que viene con un programa es parte integrante del mismo, igual que los comentarios del código fuente, como reconoce, por ejemplo en España, la Ley de Propiedad Intelectual. Dado este nivel de integración, parece lógico que a la documentación se le apliquen las mismas libertades y que evolucione de la misma manera que el programa: toda modificación que se haga de un programa requiere un cambio simultáneo y consistente en su documentación.

La mayor parte de esta documentación suele estar codificada como ficheros de texto sin formato, ya que se pretende que sea universalmente accesible con un entorno de herramientas mínimo, y (en el caso de los programas libres) suele incluir una pequeña introducción al programa (`README` o `LEEME`), instrucciones de instalación (`INSTALL`), alguna historia sobre la evolución pasada y el futuro del programa (`CHANGELOG` y `TODO`), autoría y condiciones de copia (`AUTHORS` y `COPYRIGHT` o `COPYING`), así como las instrucciones de uso. Todos ellos, menos la autoría y las condiciones de copia, deberían ser libremente modificables a medida que el programa evoluciona. A la autoría sólo se le deberían añadir nombres y créditos, pero sin borrar nada, y las condiciones de copia sólo deberían modificarse si las mismas lo permiten.

Las instrucciones de uso acostumbran a estar codificadas en formatos más complejos, ya que suelen ser documentos más largos y más ricos. El software libre exige que esta documentación pueda ser modificada fácilmente, lo que a su vez obliga a usar formatos denominados *transparentes*, de especificación conocida y procesables por herramientas libres, como son, además del texto puro y limpio, los formatos de páginas de manual de Unix, TexInfo, LaTeX o DocBook, sin perjuicio de distribuir también el resultado de la transformación de esos documentos fuente en formatos más aptos para su visualización o impresión, como HTML, PDF o RTF (formatos en general más *opacos*).

Sin embargo, muchas veces se hace documentación sobre programas por parte de terceros que no han intervenido en el desarrollo. A veces es documentación de carácter didáctico que facilita la instalación y el uso de un programa concreto (HOWTO, CÓMO o recetarios); a veces es documentación más amplia, que abarca varios programas y su integración, que compara soluciones, etc., bien en forma de tutorial o bien en forma de referencia; a veces es una mera recopilación de preguntas frecuentes con sus respuestas (FAQ o PUF). Es un ejemplo notable el Proyecto de documentación Linux (http://www.tldp.org) [44]. En esta categoría podemos también incluir otros documentos técnicos, no necesariamente sobre programas, ya sean las instrucciones para cablear una red local, para construir una cocina solar, para reparar un motor o para seleccionar un proveedor de tornillos.

Estos documentos son algo intermedio entre la mera documentación de programas y los artículos o libros muy técnicos y prácticos. Sin menoscabo de la libertad de lectura, copia, modificación y redistribución, el autor puede querer verter opiniones que no desea que se tergiversen, o al menos puede querer que esas tergiversaciones no se le atribuyan; o puede querer que se conserven párrafos, como agradecimientos; o que necesariamente se modifiquen otros, como el título. Aunque estas inquietudes pueden también manifestarse con los programas en sí mismos, no se han expresado con tanta fuerza en el mundo del software libre como en el de la documentación libre.

3.3. Resumen

En este capítulo se han revisado los aspectos legales que rigen o que tienen impacto sobre el software libre. Éstos forman parte del derecho de propiedad intelectual o industrial y han sido concebidos, en principio, para estimular la creatividad recompensando a los creadores por un tiempo determinado. De todos ellos, el llamado *copyright* es el que más afecta al software libre, y convenientemente empleado, sirve para asegurar su existencia en forma de licencias libres.

Hemos podido ver la importancia que tienen las licencias dentro del mundo del software libre. Asimismo, hemos presentado la gran variedad de licencias existentes, su motivación, sus repercusiones y sus ventajas e inconvenientes. En definitiva, podemos decir que la GPL trata de maximizar las libertades que

tiene el usuario del software, tanto si lo recibe directamente de su autor como si no, mientras que las licencias de tipo BSD lo que hacen es maximizar las libertades del modificador o redistribuidor.

A la vista de lo que se ha comentado en este capítulo, se deduce que es muy importante decidir pronto qué licencia va a tener un proyecto y conocer detalladamente sus ventajas e inconvenientes, ya que una modificación posterior suele ser muy difícil, sobre todo si el número de contribuciones externas es muy grande.

Para finalizar, queremos hacer hincapié en el hecho de que el software libre y el software propietario se diferencien de manera estricta única y exclusivamente en la licencia con la que se publican los programas. En próximos capítulos veremos, sin embargo, que esta puntualización meramente legal puede tener consecuencias –o no– en la manera como se desarrolla el software, dando lugar a un nuevo modelo de desarrollo que se diferencie en mayor o menor medida, según el caso, de los métodos de desarrollo "tradicionales" utilizados en la industria del software.