Software libre: licencias y propiedad intelectual

Jesús M. González Barahona, Joaquín Seoane Pascual, Gregorio Robles

Julio de 2004

Este artículo constituye el texto de acompañamiento a la ponencia de Jesús M. González Barahona en el Taller sobre software libre de TECNIMAP 2004.

Sinopsis

Si los aspectos relacionados con la propiedad intelectual y las licencias otorgadas por los productores son relevantes para cualquier tipo de software, en el mundo del software libre tiene una importancia especial. Las licencias, y su interpretación en el contexto de la legislación sobre derechos de autor, son el vínculo legal directo que relaciona al usuario o al distribuidor de un programa con su productor. Además, es mucho el ruido y la desinformación que se observa cuando se oyen explicaciones sobre las implicaciones de las licencias del software libre. Por ello, entender lo que las licencias de software libre tienen en común y lo que les diferencia, lo que permiten y lo que prohíben, es básico para comprender el software libre en general, sus modelos de negocio, las posibilidades de modificación y redistribución, e incluso sus implicaciones para la industria de las tecnologías de la información y sobre la sociedad en general. Y, por supuesto, para las administraciones públicas.

1. Introducción: definición de software libre

Desde hace más de 30 años nos hemos acostumbrado a que quien comercializa un programa pueda imponer (e imponga) las condiciones bajo las que puedo usarse. Puede, por ejemplo, prohibir que se lo preste a un tercero. A pesar de que el software es el elemento tecnológico más flexible y adaptable que tenemos, puede imponerse (y es común imponer) la imposibilidad de adaptarlo a unas necesidades concretas, o corregir sus errores, sin el permiso explícito del productor, que normalmente se reserva en exclusiva estas posibilidades. Pero esta es sólo una de las posibilidades que ofrece la legislación actual: el *software libre*, por el contrario, otorga las libertades que este otro *software privativo*[1] (el tradicional hasta hace no mucho tiempo) niega.

El término software libre (o *programas libres*), tal como fue concebido por Richard Stallman en su definición[2], hace referencia a las libertades que puede ejercer quien lo recibe. En concreto, estas libertades pueden resumirse en las siguientes cuatro:

1. Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.
2. Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.
3. Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos.
4. Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente.

[1] En este texto utilizaremos el término *software privativo* para referirnos a cualquier programa que no puede considerarse software libre, de acuerdo con la definición que se ofrece más adelante.

[2] *"Free Software Definition"*, <http://www.gnu.org/philosophy/free-sw.html>

El mecanismo que se utiliza para garantizarlas, de acuerdo con la legalidad vigente, es la distribución mediante una cierta licencia, como veremos más adelante. En ella el autor plasma su permiso para que el receptor del programa pueda ejercer esas libertades, y también restricciones que pueda querer aplicar (como dar crédito a los autores originales en caso de redistribución). Para que la licencia sea considerada como libre, estas restricciones no pueden ir en contra de las libertades mencionadas.

Así pues, las definiciones de software libre no hacen ninguna referencia a que pueda conseguirse gratuitamente: el software libre y el software gratuito son cosas bien distintas. Sin embargo, dicho esto, hay que explicar también que debido a la tercera libertad, cualquiera puede redistribuir un programa sin pedir contraprestación económica ni permiso, lo que hace prácticamente imposible obtener grandes ganancias simplemente por la distribución de software libre: cualquiera que lo haya obtenido puede a su vez redistribuirlo a precio más bajo, o incluso gratis[3].

Desde un punto de vista práctico, hay varios textos que definen más precisamente qué condiciones tiene que cumplir una licencia para ser considerada como de software libre. Entre ellas, destacan por su importancia histórica la definición de software libre de la Free Software Foundation, las directrices de Debian para decidir si un programa es libre, y la definición de la Open Source Initiative del término *"open source"*, muy similar a las anteriores.

En los apartados siguientes se expondrá cuáles son los fundamentos legales de estas licencias, cuáles son las principales en el mundo del software libre, y algunos otros aspectos legales relacionados con estos temas.

2. Regulación legal de la producción y distribución de software

El término *propiedad intelectual* tiene varias acepciones según el contexto y quién lo utiliza. Hoy día se utiliza en muchos foros para agrupar distintos privilegios que se otorgan sobre bienes intangibles con valor económico. Entre ellos podemos destacar los de *copyright* (derechos de autor) y similares, que protegen de la copia no autorizada los 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 la más coordinada 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.

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 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, investigación y desarrollo. Para armonizar ambos derechos, la propiedad intelectual es temporal, caducando cuando ha cumplido su función de promoción.

Pero la caducidad no es la única característica que diferencia la propiedad intelectual de la ordinaria. 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 que el robo, que sí priva del objeto al poseedor original. La copia sí puede perjudicar al propietario, ya que le 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.

Sin entrar en detalles que quedan fuera del objeto de este artículo, el software libre propone un punto de equilibrio nuevo en este ámbito, primando los beneficios de la copia y la innovación incremental frente al control exclusivo de una

[3] A pesar de que cualquiera puede comercializar un programa dado a cualquier precio, y eso hace que teóricamente el precio de redistribución tienda hacia el coste marginal de copia, hay modelos de negocio basados precisamente en vender software, porque son muchas las circunstancias donde el consumidor está dispuesto a pagar si recibe ciertas contraprestaciones, como por ejemplo una cierta garantía, aunque sea subjetiva, sobre el software que recibe, o un valor añadido en forma de selección, actualización y organización de un conjunto de programas.

obra por su autor.

3. Derechos de autor y licencias de software libre

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 en el momento de publicación de la obra con ámbito casi universal. Modernamente se ha extendido a los programas de ordenador y (en algunas áreas geográficas) a recopilaciones de datos.

La ley de la 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 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 dan derecho a explotar económicamente la obra y pueden ser cedidos total o parcialmente, de forma exclusiva a no, a un tercero. Los derechos morales son vitalicios o indefinidos, mientras que los patrimoniales tienen una duración bastante larga (70 años después de de la muerte del autor, si es una persona física, en el caso de la ley española).

La cesión de derechos se especifica por 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. Estas licencias estipulan lo que los usuarios pueden hacer con el programa en cuanto a uso, redistribución, modificación, etc., y en qué condiciones. Es la legislación sobre propiedad intelectual la que impide que legalmente se pueda, por ejemplo, vender copias de un programa que tenemos (aunque lo hayamos comprado). Por ello, el modelo de software privativo es posible porque los estados han puesto en vigor una legislación que está diseñada para ello.

Estrictamente hablando, la situación con respecto a los programas libres no es muy diferente: también se distribuyen bajo licencia. Lo que que les diferencia es precisamente qué permite esa licencia, como vimos anteriormente. En el caso de las licencias de programas libres, que hemos visto que no restringen precisamente el uso, la redistribución y la modificación, lo que pueden imponer son condiciones a 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 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 condiciones y/o restricciones que imponen las licencias sólo pueden ser precisadas por los propios autores, que según la normativa de propiedad intelectual son los propietarios de la obra. En cualquier caso, la propiedad de la obra será de los autores, ya que la licencia no supone transferencia de propiedad, sino solamente derecho de uso y, en algunos casos, de distribución. Por ello, los autores de software libre en ningún caso renuncian a la propiedad de su creación, sino que permiten a terceros que la usen, la modifiquen y la redistribuyan.

Un tema todavía relativamente abierto es la licencia que cubre 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 la licencia del proyecto, 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 de estas contribuciones.

4. Tipos de licencias de software libre

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 otra, la mayoría de los usuarios prefieren referirse a una licencia ampliamente conocida que leerse y analizar licencias completas.

Es posible dividir las licencias de software libre en dos grandes familias. Una de ellas 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 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 otra 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, ya que le permite hacer casi lo que quiera con él (en términos de condiciones de futuras redistribuciones), el segundo lo hace en la libertad de cualquiera que potencialmente pueda recibir algún día un trabajo derivado del programa (ya que obliga a que las sucesivas modificaciones y redistribuciones respeten los términos de la licencia original).

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[4].

5. Licencias robustas

Las licencias robustas, llamadas habitualmente en inglés *copyleft* tratan de garantizar las libertades que otorga el autor no sólo a quien recibe el programa directamente de él, sino también a quienes los reciben, más adelante en la cadena de distribución, de otras terceras partes. La primera de este tipo de licencias fue la GNU GPL.

La *Licencia Pública General del proyecto GNU* (más conocida por su acrónimo en inglés GPL) es con diferencia la licencia más popular y conocida de todas las del mundo del software libre. Su autoría corresponde a la Free Software Foundation (FSF, 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 a veces se traduce 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 las fuentes. Asimismo, está permitido realizar modificaciones sin restricciones. Sin embargo, sólo se puede redistribuir código licenciado bajo GPL de forma integrada (por ejemplo, mediante enlazado o *linkado*) con otro código si éste tiene una licencia compatible[5]. Esto ha sido llamado efecto *viral* (aunque muchos consideran esta denominación como despectiva) de la GPL, ya que código publicado una vez con esas condiciones nunca puede cambiar de condiciones.

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 hecho privativo. Es más, ni ese programa ni modificaciones al mismo pueden ser publicadas con una licencia diferente a la propia GPL. Como ya se ha dicho, los partidarios de las licencias tipo BSD ven en esta cláusula un recorte de la libertad, mientras que sus seguidores ven en ello 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 tipo BSD lo hacen para los desarrolladores. Nótese, 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, ya que obliga a sus competidores a publicar sus modificaciones (mejoras, correcciones, etc.) en caso de que redistribuyan su software, mientras que con una licencia tipo BSD éste no tiene por qué ser el caso.

Otras licencias robustas que puede resultar interesante comentar son:

[4]Puede ver recopiladas y comentadas tanto licencias consideradas libres como licencias consideradas no libres desde el punto de vista de la FSF en <http://www.gnu.org/licenses/license-list.html>. El punto de vista filosóficamente diferente de la *Open Source Initiative* se refleja en su listado<http://www.opensource.org/licenses/>.

[5]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. Por ejemplo, la licencia BSD actual es compatible, pero la de Apache, que exige que se mencione explícitamente en los materiales de propaganda 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 ambas.

- Licencia LGPL

  La *Licencia Pública General Menor del proyecto GNU* (comúnmente conocida por sus iniciales en inglés LGPL) es la otra licencia de la Free Software Foundation. Pensada en sus inicios para su uso en bibliotecas (la L en sus comienzos 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 la integración con cualquier otro software sin prácticamente limitaciones.

- Licencia de Sleepycat

  Es la licencia bajo la que la empresa Sleepycat distribuye sus programas (entre ellos el conocido Berkeley DB). Obliga a ciertas condiciones siempre que se redistribuye el programa o trabajos derivados del programa. 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

  Es la licencia bajo la que se distribuye eCos, un sistema operativo de 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

  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 el programa 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 el código fuente. La Affero GPL tiene una cláusula que obliga 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 fuente, cualquier usuario puede obtenerlo, y además esa *redistribución* está sometida a las condiciones de la licencia. La Free Software Foundation está considerando incluir provisiones similares en la versión 3 de su GNU GPL.

- IBM Public License Version 1.0

  Es una licencia que permite la redistribución binaria de trabajos derivados sólo si (entre otras condiciones) se preve 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 por obligar 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

  Ejemplo de licencia libre con origen en una empresa. Es una evolución de la primera licencia libre que tuvo el Netscape Navigator, y en su momento fue muy importante por ser la primera vez que una empresa muy conocida decidió distribuir un programa bajo su propia licencia libre.

6. Licencias permisivas

Las licencias permisivas, a veces también llamadas liberales o minimalistas, no imponen prácticamente ninguna condición sobre quien recibe el software, y sin embargo le dan permiso de uso, redistribución y modificación. Este enfoque, desde un punto de vista, 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 de que una vez recibido el programa por alguien, se sigan garantizando las mismas libertades cuando ese programa se redistribuye. De hecho, estas licencias típicamente 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 el punto que en muchas ocasiones se refieren las licencias permisivas como *licencias 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 dar crédito a los autores, mientras que permite tanto la redistribución binaria y la de los fuentes, aunque no obliga a ninguna de las dos en ningún caso. Asimismo se da permiso para realizar modificaciones y ser integrada con otros programas casi sin restricciones.

Las licencias permisivas son bastante populares, y existe toda una familia con características similares a la BSD: XWindow, 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 había sido pagado 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 pueda 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, ven en ella la máxima expresión de la libertad y argumentan que, a fin de cuentas, se puede hacer (casi) lo que se quiera con el software.

La mayoría de las licencias permisivas son una copia calcada de la original de Berkeley, modificando todo lo referente a la autoría. En algunos casos, como la licencia del proyecto Apache, incluyen alguna cláusula adicional, como la imposibilidad de llamar las versiones redistribuidas de igual manera que el original. Todas suelen incluir, como la BSD, la prohibición de usar el nombre del propietario de los derechos para promocionar productos derivados.

A continuación se describen brevemente algunas licencias permisivas:

- Licencia de X Window versión 11 (X11). Es la licencia usada para la distribución del sistema X Window, el sistema de ventanas más ampliamente usado en el mundo Unix, y también en entornos GNU/Linux. Es una licencia muy similar a la licencia BSD, que permite redistribución, uso y modificación prácticamente sin restricciones. A veces, esta licencia es llamada *"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. Esta licencia (habitualmente llamada *"ZPL"*) es usada para la distribución de Zope (un servidor de aplicaciones) y otros productos relacionados. Es una licencia similar a la BSD, con el interesante detalle de prohibir expresamente el uso de marcas registradas por Zope Corporation.

- Licencia de Apache. Licencia bajo al 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* (en el dominio público, o del común). La principal consecuencia de esta declaración es que el autor renuncia a todos sus derechos sobre el programa, y por lo tanto puede modificarse, redistribuirse, usarse, etc. de cualquier manera. A efectos prácticos, es muy similar a que el programa esté bajo una licencia tipo BSD.

7. Distribución bajo varias licencias

Hasta este punto se ha dado por supuesto que cada programa se distribuye bajo una única licencia en la que se especificaban 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 se difieren en la licencia. Como veremos, en la mayoría de los casos, esto se traduce en que dependiendo 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 GPL podían hacer uso de una versión de Qt GPL, mientras que si se quería integrar con programas con licencias incompatibles con la GPL (por

ejemplo, licencis privativas), debían comprarles una licencia especial. Esta solución satisfizo a todas las partes, y hoy día KDE es considerado 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.

8. Cláusulas de no responsabilidad

Todas las licencias de software libre (igual que las de software privativo) 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 el programa en cuestión se ejecuta.

Es importante también destacar dos características especiales de las cláusulas de no responsabilidad en el caso del software libre:

- En la mayoría de los casos el usuario no recibe un programa libre como resultado de una transacción comercial 8directa o indirecta) con el productor. Por ello, parece razonable que el autor no tenga responsabilidad sobre el uso que pueda hacer del programa cualquiera si no ha tenido ningún tipo de relación comercial con él, ni contraprestación económica.

- Cualquiera puede, si quiere, proporcionar servicios con cualquier tipo de garantía sobre cualquier programa libre. Puede hacerlo no sólo porque el programa esté disponible para su redistribución, sino también porque puede inspeccionarse y modificarse (por ejemplo, para resolver un error que presente), por lo que una empresa especializada podría decidir, después de estudiar y estabilizar un programa a su gusto, ofrecerlo de forma garantizada a sus clientes (que quizás estarían dispuestos a pagar por ello una cantidad que permita a esa empresa sufragar el estudio y estabilización).

9. Patentes de software

La alternativa al secreto comercial es la patente. A cambio de un monopolio comercial normalmente de 20 años, y de un determinado coste económico (las patentes hay que solicitarlas en la Oficina correspondiente, y renovarlas anualmente), se revela públicamente una *invención* de forma que sea fácilmente reproducible. Con las patentes 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 más voces que afirman que la dificulta, al menos en ciertos sectores (y especialmente en aquellos basados en innovación incremental) ya sea porque opinan que el sistema está mal implementado, ya porque creen que es perverso en sí mismo.

Qué se considera una invención ha ido variando con el tiempo, existiendo grandes presiones para ampliar la cobertura del sistema, incluyendo 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, y las presiones de la OMPI pretenden eliminar la necesidad de que el invento tenga aplicación industrial (exigido por ejemplo en la legislación europea) 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, siendo además el más beligerante para que otros países adopten sus estándares, sin recordar que él mismo se negó a aceptar las patentes extranjeras cuando era un país subdesarrollado.

Una vez obtenida una patente (tras un proceso de examen en la Oficina de Patentes), 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 la sitúan en una posición que les permite ahogar cualquier competencia. Dada la facilidad para colocar patentes sobre soluciones triviales o de muy amplia 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 de más recursos) logrará patentar su invento, perdiendo las otras 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, puede serlo en muy breve plazo.