UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS JURÍDICAS Y SOCIALES
DEPARTAMENTO DE DERECHO COMERCIAL

PROTECCIÓN DE LOS PROGRAMAS COMPUTACIONALES: DERECHO DE AUTOR Y CONCEPTO DE LICENCIA DE PROGRAMAS.

MEMORIA PARA OPTAR AL GRADO DE LICENCIADO EN CIENCIAS JURÍDICAS Y SOCIALES

Autor: Daniel Alejandro Rojas Carreño

Profesor Guía: Rodrigo Cooper Cortés

Año: 2020

Índice

Resumen:

La información y sus procesos relacionados han sido siempre relevantes para la humanidad, y ambos aspectos se hallan en un contexto particular donde los sistemas computacionales y redes de estos son protagonistas. Siendo máquinas diseñadas y definidas por y para seres humanos, estos dispositivos conservan y comunican información por y para nosotros. De entre esta información definitoria más importante social y económicamente, están los programas: órdenes o instrucciones que permiten convertir a un computador en una máquinas altamente versátiles. El propósito de este ensayo consiste en determinar el tratamiento particular que realiza el derecho de autor chileno respecto de los programas bajo una filosofía lo más consistente con su naturaleza real: información.

Palabras clave: *información, computadores, software, programa, propiedad intelectual, derecho de autor.*

1. Introducción.

Al analizar la historia canónica de los computadores o sistemas informáticos que conocemos actualmente como tales, resulta patente que la tendencia en su evolución ha obedecido al potenciamiento constante, en rapidez y efectividad, de las capacidades de procesamiento de información.

Las primeras computadoras consideradas como tales, utilizadas para fines eminentemente militares y científicos en el período de la Segunda Guerra Mundial, ofrecían funcionalidad eminentemente matemática y no tenían una maleabilidad funcional comparable a la de un computador moderno[1], aunque *computar* es un concepto que se traduce en inglés y español como *calcular*. La diferencia entre un ábaco, una calculadora ordinaria y un computador, todos artificios humanos considerados canónicamente como parte de un mismo filo tecnológico (Ifrah, 2001)[2] (Falk, 2018)[3] (Ungericht, 2013)[4] está, fundamentalmente, en la posibilidad de agregar valor a sus abstractas funciones basales (ej. matemáticas y lógicas). Sella esta identidad de un computador como una máquina procesadora, el hecho de que a partir de la década de 1960, cuando el avance tecnológico de los computadores asomaba que se extenderían por toda la sociedad, ya era posible ubicar a estos artificios como los protagonistas de una era humana especial relacionada a la información: una era en que los procesos de conservación y difusión de esta se realizaban de maneras cada vez más rápidas y efectivas hasta niveles "automáticos", de ahí las ideas de "era informática" a que aludía Karl Steinbuch en su obra homónima de 1957 y de unos bien particulares *sistemas informáticos.*

Producto de que los computadores modernos son dispositivos electrónicos digitales, es posible utilizar el concepto de "información digital" para aludir a la información que trata y procesa un computador. Según *The Linux Information Project* ("LINFO"), la información digital que se monta sobre el aparataje electrónico digital en que consiste un computador se conoce como software, por ser inmaterial (por ser

[1] La insigne *Electronic Numerical Integrator And Computer* (ENIAC) operativa desde 1946 para calcular trayectoria de proyectiles se "programaba", si es que puede llamarse así, mediante la redisposición de sus cables. Se consideraba "de propósito general" ya que podía sumar, multiplicar y hacer potencias.

[2] Ifrah, G. (2001). *The universal history of computing: from the abacus to the quantum computer.* New York: John Wiley & Sons Inc. p. 11.

[3] Falk, J. (2018). *The Modern Epoch And The Emergence Of The Modern Calculator.* Obtenido de *Metastudies:* http://metastudies.net/pmwiki/pmwiki.php?n=Site.TheModernEpochAndTheEmergenceOfTheModernCalculator

[4] Ungericht, R. (2013). *From Calculators to Computers - A Brief History.* pp. 3-18.

información) y cambiante (ej. *preemtive multitasking*), contrasta con el aparataje electrónico en sí o hardware, caracterizado por ser físico y estable (The Linux Information Project, 2005 - 2006)[5].

Dentro del software, vale decir, dentro de todo el catálogo de información digital que procesa un computador físico (ej. bases de datos digitales), se ubican los programas, que pueden definirse como "un conjunto de instrucciones para ser usadas directa o indirectamente en un computador a fin de efectuar u obtener un determinado proceso o resultado"[6]. Los programas son la *especie de software* por antonomasia hasta nivel de identidad conceptual al relacionarse a las funciones del computador, esto es, la utilización de todo el potencial de procesamiento para resultados concretos. Por esta última situación, no nos debe sorprender el hecho de que exista un verdadero "mercado" de programas o industria de software, inserto quizás en una industria más general denominada como "tecnologías de la información", todas situaciones que implican que se ha asignado valor real de manera espontánea a la información y a sus procesos relacionados.

Aún si ambos ámbitos están íntimamente relacionados al componer un único ámbito conocido como computador, es posible distinguir al menos a partir de la década de 1980, la maduración de la industria de software, lo cual significó un margen de independización de esta con respecto a la industria o mercado de los ámbitos físicos o hardware y las características clave que ha venido consolidando desde entonces. Esta independización fue gradual y se servía directamente del mayor acceso al público de los ámbitos físicos, margen mínimo de interrelación entre las industrias de hardware y software, tal situación pudiendo datarse tan pronto como el año 1951 cuando salió la *Ferranti Mark 1*, el primer computador sacado al comercio. Otro evento notable, jurídicamente relevante y que amplió dicho margen de independización, fue la demanda de libre competencia iniciada por el gobierno de Estados Unidos contra la *International Business Machines, Corp.* (IBM), a esa altura líder indiscutida en computación, y que en el año 1969 dio paso a la desagregación de su oferta de productos de hardware y soluciones de software, situación que amplió las oportunidades comerciales para interesados en la segunda.

---

[5] The Linux Information Project. (2006). *Software Definition*. Obtenido de *LINFO:* http://linfo.org/software.html

[6] Art. 5 letra t de la Ley 17.336. Esta definición destaca que un programa será tal por su aptitud para instruir a la máquina, aún si tal aptitud es "indirecta", como el caso del código fuente, el cual debe atravesar una etapa de procesamiento previa para tal efecto (compilación).

La maduración de la industria de software en la forma dicha implica una circunstancia muy singular: personas día a día buscan únicamente *intercambiar información* y también, en gran medida, servicios relacionados a esta (ej. procesamiento, almacenamiento, difusión). Esta circunstancia trae implicancias especiales que conviene advertir. Hay que considerar que en nuestra "era informática", con respecto a un creciente número de personas a nivel mundial, la información puede copiarse prácticamente ad infinitum, con exacta fidelidad, y a su vez, puede difundirse sin barreras espaciales relevantes, todo a velocidades risibles. Tal como advierte Sono (2008)[7], el problema económico ("escasez") no afecta a la información cuando esta es analizada autónomamente. Nos podemos aventurar y sostener que la información, al día de hoy y al menos autónomamente considerada y, por esta situación de oferta infinita, no tiene valor económico. Sucede que en las industrias donde el producto o servicio es la información, los actores requieren de generar una suerte de escasez artificial, por diversos medios (ej. sistemas de propiedad intelectual, contratación restrictiva), pero también es cierto que pueden basar su negocio ofreciendo servicios relacionados a la información como el caso de empresas que divulgan gratuitamente la información y muchas veces de maneras muy transparentes o abiertas, obteniendo ganancias por otras operaciones (ej. mantenimiento del programa).

Por último, debe advertirse que hardware y software siempre presentarán una interdependencia, ya que el uno y el otro, autónomamente considerados, devienen en inútiles (The Linux Information Project, 2005 - 2006). Es más, es muy reconocido que en el desarrollo y el detalle de los sistemas computacionales así como las redes de estos, rige el llamado principio de interoperabilidad: un conjunto de exigencias o ideales que inspiran la forma en que se debe intercambiar la información relacionada en torno a maximizar los procesos relacionados, especialmente su procesamiento[8]. Este principio de interoperabilidad, una especie de autoregulación en la industria, a veces también recibe recepción legal expresa, con diversos alcances, como el caso del art. 71 Ñ letra b de nuestra Ley 17.336, el considerando

---

[7] Sono, H. (2008). *The Applicability and Non-Applicability of the CISG to Software Transactions. C. B. Andersen, U. G. Schroeter (eds.). Sharing International Commercial Law across National Boundaries: Festschrift for Albert H. Kritzer Wildy, Simmonds & Hill Publishing, London, 2008,* p. 512-526. Obtenido de *Institute of International Commercial Law: https://www.cisg.law.pace.edu/cisg/biblio/sono6.html.*

[8] El principio de interoperabilidad afecta cada componente y subcomponente del sistema computacional (hardware y software en general, así como instancias más particulares de uno u otro, ej. memoria y CPU, sistema operativo y un programa aplicación) y puede referir a cualquier las relación posible (hardware con software, hardware con hardware, software con software, sistemas computacionales completos con respecto a otros sistemas computacionales). El principio de interoperabilidad desemboca en dos fenómenos prácticos importantes: adopciones de estándares de interoperabilidad y la presencia de necesarios ámbitos de interconexión entre cada componente.

(10) de la Directiva 2009/24/EC de la Unión Europea sobre Protección Legal de los Programas Computacionales, y la Sección 1201 (f) (4) de la *Digital Millennium Copyright Act* de Estados Unidos en relación al número (1) de la misma Sección, entre otras.

El estado actual de los computadores, las redes de estos y los procesos por los que atraviesa la información en ambos casos, es fascinante, ya que se trata de dispositivos ampliamente extendidos en la sociedad. Coloquialmente, con el término computador se alude a sistemas informáticos completos y autónomos, de propósito general, accesibles monetariamente y de conveniente uso personal (ej. *notebooks*, *smartphones*). También existen sistemas completos y autónomos diseñados en función de usos de distinta naturaleza a la meramente personal y, por ende, mucho menos accesibles (ej. servidores), algunos de estos con increíbles capacidades de procesamiento y tamaños (ej. terminales, mainframes, supercomputadoras). También existen verdaderos sistemas computacionales destinados a realizar tareas específicas dentro de sistemas eléctricos más complejos o grandes, vale decir, sistemas no completos ni autónomos ni de propósito general, que se denominan sistemas embebidos, los cuales, pese a no ser reconocidos generalmente como tales, representan el grueso de los computadores existentes, presentes en dispositivos como cafeteras, microondas, juguetes electrónicos, implementos médicos, ascensores, y cuyo fundamento está en que es más eficiente y barato implementar sus funciones como sistemas computacionales (aquí entendido como chip procesador programable) que disponerlas de manera prefijada en el circuito físico mismo (*custom chips*).

1.1. La Información de Software.

Software alude a la información montada en un computador físico, vale decir, información digital y que incluye a los programas y también a toda la otra posible información. Si bien este es el concepto técnico de software a efectos de este ensayo se utilizará como sinónimo de programa.

Que sea información digital implica que esta información sigue la estructura propia de los sistemas digitales (ej. símbolos utilizados). La estructura básica del software entonces está generalmente representada por 1s y 0s (números binarios) y en sus aspectos funcionales debe seguir irrestrictamente el repertorio de instrucciones del procesador, este constituyéndose por operaciones matemáticas y lógicas básicas. Puede comprenderse esta forma esencial de representar la información de software como lenguaje de la máquina.

Sobre esta base, se estructura una ordenación un poco más abstracta y, a esta altura, verdadero estándar: archivos computacionales. Un archivo puede definirse como un lugar o contenedor lógico en un dispositivo de almacenamiento y que alude a un conjunto ordenado de bits, vale decir, un conjunto de información digital definido. Cada archivo aparece para el usuario como un único y particular bloque de información y se encuentra claramente determinado representándose tal circunstancia con la ayuda de interfaces gráficas como ventanas, carpetas, íconos, etcétera (The Linux Information Project, 2006)[9].

Como software, los programas merecen mención aparte. La fase inicial de todo programa es su ingreso al computador mediante escrituración. Un programa en su faz escrita se denomina código. El código escrito por un humano (programador), vale decir, la versión originaria de todo programa, recibe el nombre de código fuente (*source code*). El código fuente utiliza en casi todos los casos caracteres y patrones entendibles por uno (ej. sistema alfanumérico, similitud con lenguas naturales) y según un lenguaje de mucho mayor abstracción con respecto a los incomprensibles detalles del lenguaje de la máquina. Debido a que solo el código en lenguaje de la máquina puede ser ejecutado (*machine code*), el código fuente pasar normalmente por un proceso de traducción, disponiéndose programas especiales a ese efecto (ej. compiladores, entornos de desarrollo, interpretadores). Previo a que un programa permita operar efectivamente al computador, este debe ser cargado del almacenamiento a la memoria (*load time*), y también debe atravesar, si corresponde, un cúmulo de otras fases que permiten preparar el programa para su correcta ejecución (ej. enlazamiento o *linking*, instalación). Un programa en ejecución (*run time*) recibe el nombre de *proceso* o tarea, y debe comprenderse como una instancia separada del programa que lo vio nacer (ej. puede ejecutarse un mismo programa varias veces y de manera separada).

1.2. Complejidad del software.

Los computadores y cada uno de sus componentes, como máquinas procesadoras, transforman los componentes más básicos de la información de software usualmente en cantidades increíbles, al punto que pueden, por ejemplo, permitir representarla convenientemente en atractivas interfaces gráficas, proveer funcionalidades mucho más interesantes y prácticas que las meramente lógicas y matemáticas, y comunicarla a distancias imposibles, todo a gran velocidad. El acceso generalizado a los computadores

[9] The Linux Information Project. (2006). *Files: A Brief Introduction*. Obtenido de *LINFO:* http://www.linfo.org/file.html

y redes de estos (ej. sistemas embebidos, internet) también ha conllevado a que la información de software haya explotado en cantidad. En cuanto a los programas, además de su clara explosión en gruesa cantidad, es posible vislumbrar una sofisticación y complejizació[10] notable de las funcionalidades ofrecidas a lo largo de los años.

Precisamente, y en cuanto a los programas, según el informático Brian Kernighan "controlar la complejidad es la esencia de la programación". El remedio para que la explosión y complejización de la información de software y programas no nos sobrepase y permita que su tratamiento informático agregando "valor agregado" está, además del seguimiento de estándares de interoperabilidad, en comprender y disponer su diseño de maneras particulares. Esto no solo se refiere a que un programa determinado pueda estar distribuido en distintos archivos, sino a que la estructura interna de un programa determinado, en la práctica, se entiende como un conjunto de *componentes ("módulos", "funciones")* bien definidos que luego operan de manera interconectada[11], circunstancia que evidencia la importancia de la estructura de los programas (Raymond, 2003)[12]. Solo para efectos de este ensayo, adelantándose que se trata de un concepto de relevancia manifiesta, y haciendo un guiño a la *rule of modularity a que* alude el insigne programador Eric Raymond, esta práctica se comprenderá como *diseño modular*.

Por último, a esta altura de la industria, las empresas y entes activos realizan sus actividades en entornos disciplinados y eficientes, con metas claras y bajo paradigmas de programación, metodologías de desarrollo y buenas prácticas particulares y en constante evolución, vale decir, se habla de una verdadera *ingeniería de software*.

---

[10]A modo ilustrativo y utilizando el ejemplo del programa open-source *Open Office*, un procesador de textos contiene alrededor de 9 millones de líneas de código ordenado en miles de módulos distintos, todos necesariamente interconectadas. Siguiendo el mismo razonamiento, un sistema operativo (en este caso las últimas versiones del *kernel* de LINUX) podría contener casi 30 millones de líneas de código.

[11]Expresiones de esta filosofía de diseño modular y control de la complejidad, también relacionadas al desarrollo eficiente de programas, son la utilización de bibliotecas (*libraries*), interfaces de programación (ej. *application programming interfaces* o APIs) y entornos de trabajo de software (*software frameworks*). Otra expresión, con quizás menor relevancia que en antaño, es el uso notable de documentación en el desarrollo de software: información accesoria de índole explicativa o didáctica de un programa (ej. manuales, comentarios en el código). El uso de documentación por los desarrolladores se da principalmente para facilitar la utilización del programa por parte de los usuarios, aunque también para otros desarrolladores (ej. continuadores del proyecto).

[12] Raymond, E. S. (2003). *The Art of Unix Programming. Ch.1 Philosphy.* Obtenido de *Eric S. Raymond HomePage: http://www.catb.org/~esr/writings/taoup/html/*

1.3. Clases de software.

1.3.1. Según su función principal.

A) SOFTWARE SISTEMA.

El software sistema o *system software* es aquel cuya función es ofrecer utilidades básicas a efectos de ejecutar otra clase de programas, de modo que su malfuncionamiento o desinstalación afecta el funcionamiento de un generalizado número de programas en sus funciones más básicas.

El software sistema por excelencia es el sistema operativo, que es en realidad un conjunto de programas cuya función es administrar la ejecución de otros programas.

B) SOFTWARE APLICACIÓN.

El software aplicación es aquel que no siendo software sistema, permite realizar una o más tareas específicas. Para el usuario promedio, las distintas aplicaciones, toman la forma de un *catálogo de funciones* relacionadas a una demanda particular: le permiten dar solución a uno o más problemas prácticos.

El software aplicación o simplemente aplicación, abarca un enorme número de programas, tales como antivirus, *office suites*, videojuegos, editores de imágenes, navegadores web, entre otros.

1.3.2. Según si su desarrollo persigue fines de lucro.

A) SOFTWARE NO COMERCIAL.

El software enteramente funcional y que no exige compensación alguna por su obtención ni otra forma de compensación subsecuentes (ej. obtención y primeros 30 días gratis, después se imponen pagos), se denomina *freeware* y no debe confundirse con la categoría de software libre *(free software)* que aborda las prerrogativas de los usuarios con respecto al programa y no a la presencia (o no) de un pago por obtener el programa.

B) SOFTWARE COMERCIAL.

Es el software desarrollado con fines lucrativos. Debe advertirse que por la práctica actual de la industria no es posible determinar la comercialidad por el simple hecho de que se cargue o no un precio por la adquisición, vale decir, la mera gratuidad en un producto o servicio de software aislado. Es común la

utilización de modelos complejos con base gratuita, pero que de una u otra manera incentivan a realizar pagos al desarrollador en algún momento. Estos modelos involucran software que debe considerarse comercial, al utilizar la gratuidad como instrumento para un fin abiertamente comercial, y no comprenden auténtico *freeware*.

1.3.3. Según si otorga protección jurídica de derecho de autor.

Tras vicisitudes históricas particulares, el medio de protección jurídica de los programas por excelencia, mas no único, es el derecho de autor.

El aspecto jurídico central del derecho de autor consiste en el otorgamiento, usualmente al autor material de una obra literaria o artística, de un catálogo de derechos exclusivos, erga omnes y temporales cuyo objeto es la explotación económica de la obra (ej. programa). También se reconocen derechos de índole moral relacionados a la atribución e integridad de la obra.

A) DE DOMINIO PÚBLICO.

Es el software que no confiere derechos de autor, entendiéndose que pertenece al "intelecto colectivo" de modo que la explotación patrimonial de este software está libre de toda exclusividad.

La forma en que puede darse esta clase de obras comprende, de manera resumida, dos supuestos de hecho: por estar la obra fuera del ámbito de protección o por expiración subsecuente de los derechos conferidos (Dusollier, 2010)[13]. Virtualmente todos los sistemas internos de derecho de autor disponen la temporalidad de los derechos de autor, aunque bajo distintas fórmulas[14]. Producto de que la industria de software es relativamente joven, el software de dominio público por expiración según el paso del tiempo no es un fenómeno generalizado.

Por otra parte, la renuncia de derechos por el titular, de ser admitida por el sistema, sería la forma por excelencia del software de dominio público. Por la territorialidad de los sistemas de derecho de autor y por ser el dominio público un concepto legal definido en contraposición a la existencia misma de derechos de autor, el alcance, requisitos y efectos concretos (en caso que proceda) de la renuncia de

[13] Dusollier, S. (2010). *Scoping Study on Copyright and Related Rights and the Public Domain.* Oxford: Organización Mundial de la Propiedad Intelectual (OMPI). pp. 6-7.

[14] Sirve tener en cuenta si el país en cuestión ha ratificado el Convenio de Berna para la Protección de las Obras Literarias y Artísticas (Convenio de Berna) en atención a lo dispuesto por su art. 7 número 1.

derechos es variable de país en país (Dusollier, 2010)[15]. Sin perjuicio de lo anterior, es posible ver que los sistemas determinan la procedencia de la renuncia según tenga por objeto derechos patrimoniales o derechos extrapatrimoniales (Guadamuz, 2014)[16]. Existen países que admiten inequívocamente la renuncia de los derechos patrimoniales conferidos por el sistema de derecho de autor, como Kenia, India y Colombia, y esta situación parece ser aceptada como doctrina general, exista o no disposición legal expresa en el sistema respectivo. Respecto de la renuncia de derechos de autor patrimoniales[17], en nuestro país, el artículo 11 letra c) de la Ley 17.336 de Propiedad Intelectual ("LPICh") parece disipar toda duda, sin perjuicio de esto, el artículo 86 de la misma vuelve a generarlas. Respecto de los derechos morales la situación es un poco más uniforme y tendiente a su irrenunciabilidad, tal como dispone, por ejemplo el artículo 16 de la LPICh en concordancia al inciso final del artículo 11. Con todo, existen países que admiten la renuncia de los derechos morales y de manera expresa como el *waiver of rights* del sistema inglés[18].

Otro aspecto importante es la forma de la renuncia para que esta sea válida. Se debe tener en cuenta que por regir el llamado principio de protección automática, que implica el nacimiento de los derechos de autor por la sola creación de la obra, la renuncia debe manifestarse a través de un acto propio. Respecto de semejantes reglas, nuestro país nada dispone, aunque Guadamuz (2014)[19] sostiene la posibilidad de utilizar las reglas generales de la teoría de los actos jurídicos o las disposiciones análogas de la LPICh relativas a la cesión de derechos patrimoniales[20].

Por último, debe decirse que el software de dominio público no se identifica con el software de autor sometido a autorización de uso o licencias, de contenido especialmente laxo (ej. licencia BSD de 2 cláusulas). Estas licencias al suponer derechos de autor y no involucrar de manera inequívoca una

[15] Ibid. pp. 21-22.
[16] Guadamuz, A. (2014). *Comparative Analysis of National Approaches on Voluntary Copyright Relinquishment.* Organización Mundial de la Propiedad Intelectual. pp. 7-14; 16; 17-18; 20-22.
[17] Para Guadamuz (2014, pp. 16) la situación en Chile sería, sin más, la de una presunta "renunciabilidad del derecho patrimonial inequívoca" o expresa, al menos de la lectura aislada del art. 11 letra c.
[18] Véase el capítulo 4 sección 87(2) de la *Copyright, Designs and Patents Act* de 1988.
[19] Ob. cit. supra nota [16].
[20] Véase el art. 73 de la Ley 17.336.

renuncia, no conllevan al software de dominio público[22]. recibiendo el nombre más propio de “licencias *equivalentes* al dominio público”.

B) DE AUTOR.

Es el software que confiere derechos temporales y exclusivos a ciertas personas en conformidad al sistema de derecho de autor aplicable. El alcance del derecho de autor en los programas se investiga con exhaustividad a partir del Capítulo 5 de este ensayo. Solo interesa mencionar el concepto de software de autor a efectos de seguir con la sistematización propuesta.

Contextualizada usualmente como subclasificación del software de autor, está la distinción entre software propietario y software libre u *open-source*. Por estimarse a que esta clasificación no alude de manera exclusiva a derechos de autor y tener por objeto más en general, a la información en sí, se advierte que dicha clasificación se abordará en lo que sigue.

[22] Estas licencias especialmente laxas reciben el nombre de “licencias *equivalentes o equiparables* al dominio público” (énfasis propio).

2. El Intercambio de información.

Por software entendemos una clase particular de información que goza de valor y que por esta última razón a dado pie a una industria o mercado con un margen de autonomía sustancial y que se traduce en la presencia de ámbitos de intercambio de información y servicios relacionados a esta. El intercambio de información reconduce a la comunicación o divulgación de esta a otro, en forma de mera analogía, hablar de "adquirir", "comprar", o "vender" información. Tal como comprende Sono (2008)[23], un espía al "vender" información lo que hace es simplemente *divulgarla* a cambio de un precio.

El contexto actual más extendido en el intercambio de programas como información sigue la tendencia general del *e-commerce*: sin tiendas físicas, accediendo a internet y pagando remotamente. En cuanto al contexto jurídico, el intercambio de información entraña, como todo intercambio, un contrato. Dentro de esta primera aproximación, debe notarse que una de las expresiones específicas de la contratación en los programas, es la contratación masiva en la forma de adhesión donde el poder negocial del usuario final de la información es nulo, debiendo decidir simplemente si adhiere o no al contrato, estos llevando la denominación arbitraria de *Terms of Use* ("ToS") o *End User License Agreements* ("EULAs"). Estos contratos son usualmente extensos y complejos y, por la particular rapidez, facilidad, y carácter de adhesión, tienden a incluir cláusulas estimadas como peligrosas o abusivas para los usuarios, los cuales, usualmente, se marginan de siquiera leerlos (Hillman & O'Rourke, 2010)[24]. Es también común que estos contratos contengan cláusulas que buscan limitar o derechamente prohibir la ingeniería inversa y cualesquiera actos relacionados a descompilar el código[25], actos que de otra manera están usualmente permitidos, al menos, por los sistemas de derecho de autor que pueden también regir en el programa (ej. art. 71 Ñ letra b) de la LPICh) (Heath, 2005)[26] (Sono, 2008)[27].

---

[23] Ob. cit. supra nota [7].

[24] Hillman, R. A., & O'Rourke, M. (2010). *Principles of the Law of Software Contracts: Some Highlights. Tulane Law Review*, pp. 1521-1522; 1529-1532.

[25] Se recuerda que los programas sujetos a contratos restrictivos, por regla general, se "entregan" en forma de código objeto, vale decir, en un archivo inmediatamente ejecutable (sea un programa instalador o el programa mismo), pero sin acompañar el código fuente o documentación relacionada a los detalles del funcionamiento del programa.

[26] Heath, S. A. (2005). *Contracts, Copyright, and Confusion: Revisiting the Eforceability of "Shrinkwrap" Licenses. Chicago Kent Journal of Intellectual Property*, pp. 12-20.

[27] Sono, H. (2008). *The Applicability and Non-Applicability of the CISG to Software Transactions. C. B. Andersen, U. G. Schroeter (eds.). Sharing International Commercial Law across National Boundaries: Festschrift for Albert H. Kritzer Wildy, Simmonds & Hill Publishing, London, 2008,* p. 512-526. Obtenido de *Institute of International Commercial Law: https://www.cisg.law.pace.edu/cisg/biblio/sono6.html.*

Además de la contratación en adhesión, en la industria de software la aceptación del usuario y por ende la formación del consentimiento ocurre de manera inédita. Lo usual será que la aceptación tenga la forma de un clic del interesado en una casilla de "acepto" luego de que se le ha presentado el contrato en la interfaz de usuario, acto entonces complejo denominado simplemente *click-wrap* (Kamantauskas, 2014)[28]. Al menos en el *click-wrap*, sucede que la oportunidad en que el usuario accede a los términos del contrato que rige a la información, es posterior a la obtención de la información del programa (ej. descarga), aunque generalmente anterior o concomitante al empleo o ejecución del programa (ej. instalación, incluso en la ejecución misma) lo que se ha traducido en discusiones jurisprudenciales (Heath, 2005)[29]. Otra manera asentada en que se presta el consentimiento, igual de inédita y problemática, aunque de uso más acotado, es el *browse-wrap*: los términos del contrato se presentan digitalmente, pero por no disponerse de interfaces para que el usuario manifieste su conformidad en un acto inequívoco, se entiende que el consentimiento se presta por el uso del sitio o por otro acto implícito (ej. por la descarga). Otras formas de aceptación utilizadas, menos relevantes en la actualidad, son el *shrinkwrap* (términos se acompañan en el paquete físico y de manera visible, el consentimiento se entiende formado al abrirlo, sin perjuicio de que se agregue una instancia adicional de *click-wrap* en la instalación) e incluso el necesario envío autónomo y expreso (ej. por correo electrónico) de la aceptación al desarrollador.

2.1. Conceptos básicos.

2.1.1. Software preinstalado o empaquetado (*bundled software*).

Refiere al software ya almacenado o instalado en el hardware (normalmente el computador juntamente con otros dispositivos imprescindibles para dar uso inmediato, como teclados, monitores y mouse) y que por esta situación se "adquiere" al adquirirse el primero.

2.1.2. Software OEM (*original equipment manufacturer*).

Refiere al software estándar adquirido por otra empresa, generalmente un fabricante de hardware, a efectos de que este opere como redistribuidor.

[28] Kamantauskas, P. (2014). *Formation of Click-Wrap and Browse-Wrap Contracts. Lithuanian Academic Electronic Library*, pp. 7-8.
[29] Ob. cit. supra nota [26].

Esta práctica es extendida porque la empresa redistribuidora accede a mejores condiciones de contratación (ej. licencias “al por mayor”) que los usuarios finales. Si bien económicamente, constituye la agregación de un eslabón en la cadena de producción, para el usuario final, el método OEM reconducirá a un software preinstalado.

2.1.3. Software de retail (*shrinkwrapped software o shrinkwrap*).

Refiere a las copias de software comercializadas al público general bajo fórmulas particulares.

La información de software se “entregaba” al público general mediante la venta de un dispositivo de almacenamiento que lo contenía (ej. CD-ROM), el cual se disponía en una caja, junto con documentación relevante (ej. manuales, licencias), la cual se envolvía con plástico retráctil (*shrinkwrap*).

El dispositivo así adquirido funcionaba como simple enlace físico entre el usuario y la información (verdadero objeto de la transacción), generando eso si la apariencia de que se adquiría un producto tangible.

2.1.4. Software descargado o distribuido digitalmente.

Refiere a las copias de software que se obtienen por medio de un proceso de descarga que prescinde de puntos de venta físicos y dispositivos adicionales.

Sin perjuicio de la intensa intangibilidad o abstracción de la operación, esta si supone la existencia de un computador remoto que contiene el programa: un servidor físico. La red entre el servidor y el computador del interesado se obtiene generalmente vía páginas web. El usuario hace la petición de descarga del programa que le interesa, iniciándose así el proceso de descarga que culmina con una copia del programa en su favor.

2.1.5. *Freeware, shareware, payware, y donationware.*

Son conceptos relacionados a la contraprestación obtenida por los desarrolladores de software, acotadas al pago o no de un precio.

El freeware es un programa funcional totalmente gratuito. El shareware es un programa que solo es inicialmente gratuito, luego, tras cierto lapso o evento determinado, se generan incentivos para realizar pagos al desarrollador o se desincentiva el uso gratuito, con mecanismos de mayor o menor efectividad (ej. reduciendo las funcionalidades, avisos molestos).

El payware es el software que si exige pago para su adquisición. El donationware alude al software que fija donaciones voluntarias u obligatorias (verdadero precio elegido por el usuario) como forma de contraprestación.

2.1.6. *Freemium.*

Es un concepto compuesto de las palabras *free* y *premium* y que puede traducirse al español como *gratis con recargo*. Es un sistema de negocios complejo y por el cual se ofrece un producto o servicio inicial de manera gratuita, existiendo la posibilidad para extender o potenciar su funcionalidad realizando pagos. Tiene especial importancia en la industria de software por su extensión actual generalizada, sea en productos o servicios.

2.1.7. Software como servicio (software-as-a-service)

El software-as-a-service o, por sus siglas en inglés, "SaaS", es una expresión particular de la computación en la nube (*cloud computing*)[30] para prestar servicios y que se distancia de las formas de software preinstalado, de retail, o descargado, por ser estas formas que dependen de los medios del usuario para ser utilizados (*on-premises* versus *on-demand*).

En el SaaS, el usuario, en realidad cliente, no recibe en ningún momento copia del programa ya que simplemente "accede" a un servidor administrado por otro[31], a efectos de utilizar sus recursos de cómputo (un programa) y otros servicios (ej. almacenamiento) de manera *on-demand*. El acceso se da comúnmente a través de páginas web o mediante la utilización de programas que hacen las veces de plataforma entre el cliente y el proveedor.

El SaaS se estructura por lo general en un modelo de negocios basado en pagos a intervalos constantes (suscripción) o de pago por el uso (*pay-per-use* o *pay-as-you-go*), a veces utilizando el modelo *freemium* (ej. servicio básico gratuito expandible), y por la creciente relevancia del internet, es esperable que siga creciendo en la comercialización de software, aunque se debe observar que, por la naturaleza del modelo, la comercialización de software devendría en la prestación de servicios o funciones y no en la

[30] Por *computación en la nube* aquí únicamente se alude a la utilización de redes de computadores, sean públicas o privadas.

[31] Cf. *Don't Use License Agreements for Software-as-a-Service* de David Tollen (Tech Contract Academies, 2018), disponible en línea: https://techcontracts.com/2018/06/01/dont-use-licenses-saas-contracts/, consultada el 13 de enero de 2020.

entrega de información. Por esta razón los conceptos freeware, shareware y sucedáneos no aplican al SaaS.

2.1.8. Software personalizado (*custom software*).

Es el software desarrollado a requerimiento de una persona o empresa determinada, según sus necesidades específicas y en la forma de servicios de programación. Estos servicios generalmente incluyen el desarrollo del programa (diríamos que comprenden una obligación de hacer: escrituración del código), pero pueden extenderse a la puesta en marcha, la modificación y el mantenimiento.

El *custom software* contrasta con el software desarrollado de manera predefinida para dar satisfacción a necesidades típicas de la demanda ("software estándar" o "producto de software") y que se aplica virtualmente a todos los conceptos anteriormente presentados al aquí presente. Si se considera una forma de *custom software* el prestar un servicio de *personalización* de un software estándar.

El software desarrollado desde cero o editando un programa existente para satisfacer necesidades propias se denomina *own-account software*.

2.1.9. Software libre, open-source, y propietario.

Los términos propietario, libre y open-source, todos arraigados históricamente, aluden a la forma de divulgar e intercambiar la información de software. El centro de estos conceptos no refiere tanto a la existencia de un precio u contraprestaciones por la obtención del programa en tanto información, sino que refiere a la extensión efectiva de la información divulgada (ej. solo el código objeto) y a las facultades o actos que se permiten (o prohíben) sobre el programa (ej. no es posible ejecutar el programa para fines comerciales), siempre en tanto información.

En ese sentido, a partir de la década de 1980 la industria de software consolidó para sí una fase de masividad, situación que trajo desafíos propios y un impulso de los desarrolladores encaminado a capturar de manera segura el valor económico producido, así como un impulso legislativo para proteger dicho interés legítimo. El impulso legislativo desembocó en un reconocimiento inequívoco de los programas como obras eligibles de protección por parte de los sistemas de derecho de autor e introdujo el patentamiento en los programas a la discusión.

El impulso propio de los desarrolladores apuntó al desarrollo y adopción de medidas tecnológicas de protección (ej. llaves de activación), y el asentamiento de una nueva fórmula de intercambiar la

información de software: el usuario final accede únicamente a un archivo directamente ejecutable (código objeto); luego, se restringe y ajusta sustancialmente el marco de acción del usuario sobre el programa, primero por no acompañarse el código fuente del programa (ya no es posible estudiar su funcionamiento en una forma suficientemente apta), y segundo porque se dispone un catálogo increíblemente amplio de actos prohibidos de realizar sobre del programa. Históricamente, estas maneras restrictivas de intercambio fueron un *nuevo* paradigma porque, desde sus inicios y hasta la década de 1980, aún si el desarrollo de software se hallaba desprendido de fines militares y científicos, este aún se caracterizaba darse en una cultura donde el desarrollo colaborativo y el compartimiento de la información era un valor (Hippel & Krogh, 2009)[33].

Fue un reputado *hacker* llamado Richard Stallman quien desplegó y doto de organicidad a la primera reacción seria al mencionado nuevo paradigma: el *software libre*. Al año 1985, Stallman fundaba la *Free Software Foundation* (FSF) para dotar de organicidad a su particular perspectiva y anunciando desde ya el primer proyecto de software libre (sistema operativo GNU), vale decir, un software que cumplía con sus planteamientos. Según Hippel & Krogh (2009)[34], el movimiento de software libre no gozo de reconocimiento o influencia sustancial inmediatas. A esta altura, también es posible distinguir distintas vertientes, perspectivas y otros protagonistas o actores relevantes distintos a Stallman dentro del mismo movimiento del software libre. De hecho, fue en 1998, dentro de un movimiento de software libre maduro que los programadores Bruce Perens y Eric S. Raymond dieron nacimiento al movimiento de *software open-source* (Raymond, 2000)[35], fundando la organización homónima: la Open Source Initiative (OSI). Este movimiento fue en realidad otra manera de reaccionar contra las prácticas restrictivas de divulgar la información de software, pero esta vez bajo argumentos mucho más pragmáticos, que contrastaban con los motivos fundamentalmente ideológicos del software libre. Producto de que tener un adversario común, ambos movimientos identifican a su rival como *software propietario* o *privativo*.

---

[33] Hippel, E. v., & Krogh, G. v. (2009). *Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. Organization Science 14 (2)*, pp. 209-212.

[34] Ibid.

[35] Raymond, E. S. (2000). *The Revenge of the Hackers*. Obtenido de *Eric S. Raymond HomePage:* http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-revenge/

Si se sigue la vertiente del software libre en la vertiente de su fundador, la razón para repudiar el software propietario puede elucidarse con el siguiente extracto:

"El movimiento del software libre defiende la libertad de los usuarios de ordenadores para que sean los ellos quienes ejerzan el control del software que utilizan, y no al revés (…) *Con el software libre los usuarios tienen el control del programa*, tanto individualmente como en forma colectiva. Así, controlan lo que hace el ordenador (…) *Con el software privativo el programa controla a los usuarios*, y alguna otra entidad (el desarrollador o «propietario») controla el programa. De modo que *el programa privativo da al desarrollador poder sobre los usuarios*. Eso en sí mismo es injusto; además, el desarrollador se ve tentado a maltratar a los usuarios de otras maneras (…) *La libertad consiste en ejercer el control de su propia vida"*[36].

Si se sigue la vertiente del software open-source, tomaríamos un argumento base diverso: por software entendemos netamente información, luego, la "economía de la información" es fundamentalmente distinta a la que rige en los productos tangibles (ej. no afecta el problema de la escasez) y, producto de esto, aquellas empresas y desarrolladores que disponen de un sistema de comportamiento colaborativo de información tendrán mayores ventajas competitivas con respecto a aquellos que prefieran obstaculizarlo (Raymond, 2000)[37].

Sin perjuicio de que es posible ver un cierto antagonismo entre el movimiento libre y el open-source, en la práctica, un software particular que se comprende como libre también se comprenderá como open-source y viceversa (Stallman, 1998)[38]. La comprensión efectiva de un software como libre u open-source es independiente de la denominación que el desarrollador utilice para su proyecto, del marketing utilizado o de si el software pertenece a un proyecto auspiciado o aprobado o no por la FSF o la OSI[39]. Por esta especie de identidad, en este ensayo software libre y open-source se engloban en el concepto más neutro de *Free and Open-Source Software* ("FOSS"), aunque según la retórica a destacar

---

[36]Richard Stallman (Free Software Foundation, Inc., 2015-2020). *El Software libre es ahora aún más importante*. Disponible en línea: https://www.gnu.org/philosophy/free-software-even-more-important.es.html, consultada el 20 de enero de 2020.

[37] Raymond, E. S. (2000). *A Brief Story of Hackerdom*. Obtenido de *Eric S. Raymond HomePage:* http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/

[38] Stallman, R. (1998). *The GNU Operating System and the Free Software Movement: The Free Software Sharing Community*. Obtenido de *Dulce Libertad:* https://lorenzopena.es/linux/movement.html

[39]"OSI Approved License" es una marca de certificación. "Open-source" o "software libre" no son marcas comerciales o de certificación, al menos respecto del licenciamiento de software.

(argumentos económicos pragmáticos o reivindicación de libertades) podemos apuntar a un término más preciso.

Siguiendo y explicando lo anterior, un software se comprenderá como *libre*, cuando otorgue al usuario las siguientes *cuatro libertades*[40]:

a) Libertad para ejecutar el programa como se desee y para el fin que se desee.

b) Libertad para estudiar cómo funciona el programa, lo cual implica que se debe permitir el acceso al código fuente del programa.

c) Libertad editarlo o modificarlo para que realice las tareas que se deseen.

d) Libertad para distribuir copias ("redistribución"), del original o de las modificaciones, con el propósito que se desee, y a quien se desee.

Con respecto al software *open-source*, un software será tal si se adscribe al *estándar open-source*, prescribiéndose diez directrices, que se pasan también a explicar[41]:

a) Distribución libre, en el sentido de que esta se ha autorizado de manera generalizada y no se encuentra restringido trivialmente.

b) Inclusión del código fuente, sea al transferir las copias del programa, o contemplando mecanismos razonables para obtenerlo (Perens, 1998)[42].

c) Modificación del código libre, al autorizarse de manera generalizada sin restricciones triviales, sin perjuicio del criterio siguiente. La definición permite, de manera implícita, que estas obras derivadas podrán licenciarse como se desee, incluyendo aspectos de reciprocidad (Perens, 1998)[43].

d) Integridad del código fuente del autor, en el sentido de que es válido para el autor original limitar la distribución de modificaciones a efectos de permitir distinguir estas versiones de su obra.

---

[40] Free Software Foundation, Inc (2009-2019). *¿Qué es el software libre?* Disponible en línea: https://www.gnu.org/philosophy/free-sw.es.html, consultada el 20 de enero de 2020.

[41] Véase directamente: Open Source Initiative. *Open Source Definition* (2007). Disponible en línea: https://opensource.org/osd, consultada el 20 de enero de 2020.

[42] Perens, B. (1998). *The Open Source Definition*. Obtenido de *O'Reilly Media:* https://www.oreilly.com/openbook/opensources/book/perens.html

[43] Ibid.

e) Concesión de derechos no discriminatoria contra personas o grupos, fuera de las restricciones legales no sorteables como los de materia aduanera (Perens, 1998)[44].

f) En la concesión de derechos, no debe haber discriminación según campos de actividad o según los fines según los cuales se emplea el programa, por ejemplo si buscan propósitos comerciales.

g) Los derechos deben concederse a favor de todos aquellos a quienes se les transfiera el programa, sin la necesidad de que estos deban contar o pedir autorizaciones adicionales.

h) Los derechos concedidos no se sujetan a que el programa se encuentre o forme parte de un paquete de programas particular. Según Perens (1998)[45] un "paquete de programas" en este contexto alude a un conjunto de programas independientes que se transfieren en un mismo medio físico (ej. 3 programas distintos en un mismo CD-ROM) o por un mismo acto (ej. se descarga una archivo "zip" que contiene 3 programas distintos).

i) La licencia otorgada sobre el programa no debe imponer restricciones sobre otros programas, especialmente aquellos dispuestos en un mismo "paquete" (véase el criterio anterior).

j) La licencia debe ser tecnológicamente neutral, en el sentido de que no debe imponer o presuponer una clase de tecnología, estilo o interface en particular. Esto implica que la licencia no puede discriminar bajo el criterio del uso o no de determinada tecnología (de hardware, software u otras).

2.2. Aspectos particulares del software libre y open-source.

El software libre y open-source reconduce a condiciones de licenciamiento de la información de manera no restrictivas y su concepto no alude a que exista o no una contraprestación o la persecución (o no) de un lucro. Debe tenerse en cuenta que un FOSS particular, por las prerrogativas que otorga a los usuarios, tenderá a esparcirse naturalmente[46], no siendo el precio de obtención de copias, aún si excesivo, una limitante al criterio de acceso a la información de software.

El licenciamiento FOSS es un auténtico vehículo jurídico cuyo objeto son derechos de autor, pero no producto de que su objeto sea un programa potencialmente protegido, sino por que alude a actos que

[44] Ibid.
[45] Ibid.
[46] Cfr. Free Software Foundation, Inc. (1996-2018). *Vender software libre*. Disponible en línea: https://www.gnu.org/philosophy/selling.en.html, consultada el 21 de enero de 2020.

tienen estrecha relación con los actos propios de derecho de autor: distribución (y comunicación pública para estos efectos) y modificación. Algunas licencias FOSS también incluyen licencias de patentes expresas (ej. Apache v. 2.0). Para Scott K. Peterson (2018)[47], existirían licencias de patentes, esta vez implícitas, en caso de que se licencie un programa y se permita la ejecución de este, especialmente si se especifica que se autoriza la ejecución para fines comerciales (ej. licencia MIT). El licenciamiento propietario, tiende a reservarse los distintos derechos de autor (ej. usuario no puede distribuir el programa, ni modificar) y otros de propiedad intelectual implicados, y gusta también de imponer restricciones adicionales o ámbitos no comprendidos por la propiedad intelectual (ej. se prohíbe la ingeniería inversa sobre un programa que no tiene derechos de patente relacionados, se prohíbe toda modificación y para cualquier fin).

2.2.1. Importancia.

A extensos 35 años de su nacimiento, se debe decir que el FOSS ha devenido en extrema relevancia actual. Por una parte, el desarrollo colaborativo de software está totalmente extendido en la actualidad[48] y goza de importancia para las empresas[49]en desmedro del software propietario, y parece ser sinérgico con el principio de interoperabilidad y los estándares abiertos que rige en ámbitos tan importantes como el internet y el propio desarrollo de programas (Blind & Böhm, 2019)[50].

La importancia del desarrollo de software open-source en empresas es probablemente el aspecto más notable actualmente del FOSS. Ejemplos de empresas de FOSS viables comercialmente no son aislados (ej. RedHat, Canonical, MongoDB), la mayoría abrazando quizás el hecho de que la información en si no es escasa ni debe someterse a restricciones artificiales, de modo que prefieren ofrecer servicios relacionados a la información. Recientemente Steve Hickin (CTO de Microsoft Australia) ha llegado bastante lejos, al sostener en 2019 que el propio *Microsoft* (otrora estandarte del software propietario) es una empresa comprometida con el open-source al punto de *serlo*.

---

[47] Peterson, S. K. (2018). *Why so little love for the patent grant in the MIT License?* Obtenido de *Opensource.com:* https://opensource.com/article/18/3/patent-grant-mit-license

[48] Se tienen a la vista los datos del informe *Octoverse 2019* del repositorio de proyectos de software GitHub, disponible en línea: https://octoverse.github.com/, consultada el 21 de enero de 2020.

[49] Véase *The State of Enterprise Open Source (2020)* de Red Hat, Inc, disponible en línea: https://www.redhat.com/cms/managed-files/rh-enterprise-open-source-report-detail-f21756-202002-en.pdf, consultada el 2 de marzo de 2020.

[50] Blind, K., & Böhm, M. (2019). *The Relationship Between Open Source Software and Standard Setting.* Publications Office of the European Union. pp. 41-45; 75-78.

Sucede también que el compartimiento colaborativo de la información articula una base sólida para la innovación, primero dentro de la misma empresa y pero también con respecto a la innovación general de la industria la cual se sirve de varios programas open-source (ej. LINUX, Nginx/Apache Web Server y OpenSSL operan en virtualmente todos los servidores del planeta). En ese sentido,), en el desarrollo de software es usual que se utilicen otros programas como fuente de inspiración o como piedra angular (C. Phillips, 1992)[51], algo que los millones de usuarios promedio presentes en las plataformas GitHub o SourceForge comprenden.

Jurídicamente, la importancia del FOSS es manifiesta, ya que los problemas y particularidades de este pondrán a prueba los límites, primero de las normas que regulan el siempre creciente tráfico jurídico de la información, pero, más importante a efectos de este trabajo y por la relación inherente que tienen, las normas propias de derecho de autor.

2.2.2. Licenciamiento recíproco, copyleft y compatibilidad.

En algunas formas del licenciamiento FOSS se sigue una especie de ethos en el desarrollo de programas, más o menos explícito, de manera que no se busca otorgar libertades sin más. En estas formas del FOSS se hace esperable que el sistema de cuatro libertades o el estándar open-source sea respetado por quienes se benefician del mismo, de modo de que, ahora, dichas directrices pasan a garantizarse (Moglen, 2001)[52] (Perens, 1998)[53].

En este afán, en el licenciamiento FOSS se inserta un importante eje o subclasificación: el eje de reciprocidad. Este "eje" mencionado pasa también a contextualizar el concepto de *copyleft*. El fenómeno del *licenciamiento recíproco*, podemos definirlo sin mucho traspié según el concepto de *share-alike* presente en las ampliamente utilizadas licencias *Creative Commons*: licencias que *restringen la distribución de obras derivadas, las cuales no podrán licenciarse en términos más restrictivos de los que regían a la obra original*. Según el propio Stallman (1999)[54], el *copyleft* implica que a *los licenciatarios de un programa libre se les restringe el derecho de distribución de obras derivadas,*

[51] C. Phillips, J. (1992). *Sui Generis Intellectual Property Protection for Computer Software*. Obtenido de *Berkman Klein Center:* https://cyber.harvard.edu/property/protection/resources/phillips_unedited.html
[52] Moglen, E. (2001). *Enforcing the GNU GPL*. Obtenido de *The GNU Proyect:* https://www.gnu.org/philosophy/enforcing-gpl.es.html
[53] Ob. cit. supra nota [42].
[54] Ob. cit. supra nota [38].

debiendo estos aplicar los mismos términos que regían al programa original (reciprocidad exacta) y al hacerlo permite precaver y proteger de que el código devenga eventualmente en propietario ("protectivo"), ya que si no existiera esta restricción, el código podría utilizarse en software propietario sin consecuencias ("permisivo"). Se prefiere hablar de "eje", en cuanto es posible determinar distintos grados o coordenadas. En ese sentido es posible, por ejemplo, distinguir el *copyleft* en débil o fuerte, según la extensión de obras derivadas alcanzadas por la restricción.

Sobre el *copyleft* debe advertirse que su concepto no implica el abandono (en inglés "left") de los derechos de autor, vale decir, no conduce al dominio público, sin perjuicio de que el software de dominio público es considerado un efectivo FOSS. De hecho, el *copyleft*, al tener sentido (al menos histórico) únicamente en el FOSS y, recordando que el FOSS tiene estrecha relación con los derechos de autor, es que el *copyleft* puede comprenderse como una forma de utilizar el sistema de derechos de autor o *Copyright*. Sobre estos derechos, el *copyleft* parece relacionarse más, sin perjuicio de la conceptualización de su propio fundador, al derecho de adaptación que del de distribución. En ese sentido, diríamos que un autor tiene derecho a distribuir la obra original y también a preparar obras derivadas de esta, estas las cuales, al ser una efectiva obra autónoma de la original (ej. art. 5 letra i Ley 17.336), produce o crea su propio derecho de distribución. Es posible advertir que si la restricción transaccional propia del *copyleft* refiere a otra cosa que no sea una obra derivada (ej. el programa original), se estaría vulnerando la libertad 3 y el estándar open-source primero y octavo.

Solo relacionado al tópico del licenciamiento recíproco (y por ende al *copyleft*) es que ocurre el problema de la compatibilidad de licencias. En este caso, el término compatibilidad o incompatibilidad se contextualiza en ciertos casos, relacionados al *desarrollo modular de programas*: componentes del programa sujetos a distintas licencias, libres o no, en caso de que alguna de estas imponga prohibiciones o restricciones relativas al resto del código que se incluya en el programa final. Según Stallman (2016), el programador que incluya código sujeto a una licencia libre con *copyleft* en su propio proyecto y que desee distribuir este último, debe hacerlo obligatoriamente bajo una licencia libre, en el entendido a que este programa final es una obra derivada del primero. Pero si en dicho programa se han incluido bloques de código sujetos a una licencia propietaria, al distribuir dicho programa final se estaría vulnerando el *copyleft* de la o las licencias libres a que pueden estar sujetas otros bloques del código. Este problema es, en teoría, inexistente en las licencias que no imponen restricciones tipo *copyleft*

también denominadas por esta misma razón como licencias permisivas. Un aspecto quizás curioso, es que las distintas licencias libres con *copyleft* son estimadas como formalmente incompatibles entre sí ya que es imposible cumplir al mismo tiempo con el *copyleft* de cada una.

Todo lo anterior debe entenderse sin perjuicio de las modalidades de compatibilidad expresas que comprenda la licencia (ej. el programa se sujeta a la licencia GPL v.2 o cualquier versión siguiente) o permisos expresos para "relicenciar" la modificación, usualmente extendido a un catálogo limitado de licencias autorizadas para estos efectos. En este contexto, el *relicenciamiento* alude a la licencia que regirá la modificación y que es objeto de la restricción propia que dispone el *copyleft*, por lo que, a priori y según los propios paladines del software libre, debe ser idéntica licencia (Stallman, 2016)[56]. En este último caso, debe advertirse que dichos permisos para relicenciar no solucionan el problema de la compatibilidad, ya que no son per se recíprocos entre licencias por el hecho de estar incluidos en una licencia particular y no en la otra licencia potencialmente implicada.

2.2.3. Proliferación o complejización.

Un segundo aspecto relevante del intercambio de información en el FOSS está en la *proliferación* de licencias. De los gruesos datos de la página *SourceForge,* podemos deducir que existen más de 70 licencias FOSS de uso relevante en el tráfico y esto sin contar las licencias creadas ad hoc por cada desarrollador.

Si bien en la práctica se utiliza un selecto número de licencias[57], para Scott K. Peterson (2020)[58], el FOSS si resiente los efectos de la proliferación. Por el hecho de que la filosofía FOSS implica facilitar la contribución de varias personas a un mismo proyecto[59], siendo incluso posible determinar la existencia de distintas ramas de una misma agregación o componentes, si a esto se suma el problema de la reciprocidad e incompatibilidad, se está en presencia de una circunstancia increíblemente compleja. Si

---

[56] Stallman, R. (2016-2018). *License Compatability and Relicensing*. Obtenido de *GNU Operating System:* https://www.gnu.org/licenses/license-compatibility.html

[57]Estas son: GNU GPL v.2 y GNU GPL v.3, GNU LGPL v.2, MIT, Apache v.2.0., BSD original y BSD modificada.

Fuentes: *Top Open Source License Trends* (GitHub: https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions, 2019); y *The Relationship Between Open Source Software and Standard Setting* (Blind & Bohm, ob. cit. supra, pp. 47).

[58] Peterson, S. K. (2020). *Is open source licensing broken?* Obtenido de *Opensource.com:* https://opensource.com/article/20/2/open-source-licensing

[59] Esta noción de contribuciones independientes debe relacionarse al diseño modular. Esto implica que el FOSS será, en general, una agregación de programas independientes o fusionados, y con seguridad producto de aportes de distintos autores materiales.

bien el compartimiento colaborativo autoorganizado es el contexto que otorga al FOSS el dinamismo por el que es conocido, también da pie a la *complejización*, no solo de la información del programa, sino de las reglas a las cuales se sujetan los distintos actos que pueden recaer sobre estos, vale decir, *la información de la licencia* misma. Esta situación también tendría efectos relacionados a la efectividad y plausibilidad de accionar en juicio, circunstancias así igual de enmarañadas.

2.2.4. Exigibilidad.

Los problemas de exigibilidad general de las licencias FOSS apuntan principalmente a la pregunta de si, constando de que el contenido de una cláusula abarca efectivamente derechos de autor, es posible entonces exigir su cumplimiento basado en dicho sistema y no meramente comprendiéndolo como un incumplimiento contractual. Debe recordarse que las cláusulas de una licencia FOSS más que imponer restricciones, disponen y garantizan libertades, y en general, tampoco se pacta un precio o regalía, sea por la obtención de la información, o por el ejercicio de las facultades o derechos concedidos (más propiamente, "autorizados") sobre el programa.

El caso *Jacobsen v. Katzer* (2008, Fed. Cir., "Jacobsen"), arrojó luces al respecto, sosteniendo que las acciones seguidas por el demandado en el caso concreto, *copia, modificación y distribución del programa,* se sujetan a los términos de la licencia FOSS en su aspecto propio de licencia de derechos de autor. Esto se traduce en que el licenciatario, debía de cumplir con las condiciones de licenciamiento prescritas, que en dicho caso aludían principalmente a las cláusulas 3 y 4 de la licencia en cuestión (*Artistic License* 1.0[60]) y que son regidas, por virtud de tratar actos propios de derecho de autor ("copia", "modificación", "distribución"), por dicho sistema y no otro. Esta importancia de la naturaleza del acto realizado sobre el programa (ej. modificar o ejecutarlo para propósitos personales) es un criterio muy importante para decidir las normas aplicables (ej. derecho de autor, patentes, o normas contractuales).

Debe advertirse que la contraparte basó su defensa en un argumento específico, quizás ya aceptando que el derecho aplicable es el derecho de autor: el licenciamiento FOSS no persigue fines económicos y

[60] Sobre esta licencia particular, el tribunal comprendió que "un usuario que descarga la obra (…) está autorizado para realizar modificaciones y para distribuir los materiales "siempre y cuando" cumpla con los términos restrictivos de la Artistic License (…) está *fuera del ámbito de la licencia* la modificación y distribución de los materiales sin los avisos de derecho de autor y sin hacer el seguimiento de las modificaciones de la obra original" (traducción y cursiva propia).

En ese sentido, además del contenido de los actos, la forma en que se redactan las restricciones es un aspecto importante para estar en presencia de una condición de licencia de derecho de autor y no una mera cláusula contractual.

por ende no está tutelado por los derechos de autor patrimoniales, algo que implicaría que el licenciamiento FOSS es una especie de renuncia. Desestimando fundadamente tal planteamiento[61], el tribunal sostuvo que el contenido de una licencia FOSS es generalmente obligatorio y, al menos en ciertos casos, por la vía de derecho de autor. Queda pendiente la determinación jurisprudencial del alcance de las cláusulas de licenciamiento recíproco y del *copyleft*[62], algo que quizás se responda prontamente en atención a la creciente importancia del FOSS en general.

[61]El tribunal sostuvo: "Existen beneficios sustanciales, incluyendo de tipo económico, en la creación y distribución de obras amparadas por derecho de autor bajo licencias públicas y que son muy distintos a las maneras tradicionales a través del pago de regalías (…) El *11th Circ.* ya ha reconocido los motivos económicos inherentes en las licencias públicas, incluso si las ganancias no son inmediatas (*El desarrollador del programa ha obtenido valor de la distribución bajo una licencia pública porque ha podido mejorar el programa basándose en las sugerencias enviadas por los usuarios finales (…) en tanto el programa ha mejorado, más usuarios han ido utilizando el programa, de manera que se ha ido incrementando el reconocimiento profesional del desarrollador y las chances de que el programa mejora aún más*) (…) A través de esta manera controlada de difundir información, el titular de derechos de autor gana contribuidores creativos a su proyecto open-source; por el hecho de requerir que los cambios hechos por los usuarios "rio-abajo" sean informados al titular y a otros interesados, el titular aprende sobre las funciones de su programa y aprende del conocimiento de los demás, todo lo cual puede ser utilizado para continuar con futuros lanzamientos del programa".

[62]El caso *Hellwig v. VMWare* (HansOLG, 2016) presentó este tema en juicio, pero la demanda fue desestimada en primera y segunda instancia por cuestiones procesales y sin entrar a ver el fondo.

3. Protección de los Programas.

Por la importancia de los programas, sea respecto del propio desarrollador o de los usuarios, los distintos actores se han visto en la necesidad de disponer y utilizar medios para velar por sus más diversos intereses (ej. el *copyleft* más que propósitos comerciales busca proteger el ethos de desarrollo colaborativo). Al menos desde el punto de vista de un desarrollador de software, se hace posible distinguir entre medios tecnológicos y jurídicos de protección.

3.1. Medios Tecnológicos de Protección.

Los medios tecnológicos de protección en los programas y otra clase de información digital, *technological protection measures* ("TPMs"), también se comprenden con el concepto de gestión de derechos digitales o *digital rights management* ("DRMs")[63]. En este ensayo se prefiere el término más gráfico de medida de protección tecnológica o *TPM* por sus siglas en inglés.

Siempre en el ámbito de la información digital y las problemáticas en torno a la protección de los intereses en juego, pueden definirse las TPMs como cualquier método tecnológico destinado a promover el uso autorizado o esperado por el desarrollador, de un determinado bloque de información digital[64]. Estos TPMs se desarrollan para dotar de necesaria exigibilidad, respecto de, indistintamente, derechos de autor implicados, derechos y obligaciones contractuales que puedan regir, y todo otro interés que tenga el desarrollador.

Tal como nota R. Kerr et al (2005)[65], es posible encontrar doctrinariamente, e incluso recepciones legales expresas[66], sobre una presunta clasificación canónica de las TPMs en base a su función: aquellos que controlan el *acceso mismo* a la información digital o aquellos que controlan el *uso* de esta. Sin perjuicio

[63]Para Lara & Vera (2014, ob. cit. infra nota [76], pp. 3-4), los DRM son una *clase* particular de TPM que goza de carácter digital (léase electrónico, ej. un programa o la gestión de derechos con la ayuda de redes de computadores).

[64] Se prefiere el término neutro de "bloque" ya que no es correcto aludir a un archivo computacional sin más. Es usual que un producto de software comprenda decenas o miles de archivos de distinta naturaleza (ej. ejecutables, imágenes, y *data* genérica en distintos formatos). Determinar que corresponde o no a un determinado "bloque" es una tarea que debe precisar el desarrollador al licenciar o dar acceso a la obra.

[65] R. Kerr, I., Maurushat, A., & Christian, S. T. (2005). *Technological Protection Measures: Tilting at Copyright's Windmill. Ottawa Law Review Vol. 34, No.1*, 13-22.

[66]Por ejemplo, en la DMCA de Estados Unidos, la Sección 1201 (a) refiere a TPMs que controlan el acceso a la obra (TPMs de acceso). La letra (b) de la Sección 1201 habla de las TPMs que protegen los derechos de autor tradicionales (reproducción, distribución, etcétera). En ese sentido, la letra (b) refiere a TPMs que controlen actos distintos al acceso (el acceso a la obra no es un derecho de autor tradicional) y que se relacionen a los TPMs de uso.

de esto, los autores referidos también notan que, en la práctica, los TPMs pueden tener ambas funciones, u otras distintas, siendo de hecho la regla general que así sea (R. Kerr, Maurushat, & Christian, 2005)[67].

El desarrollo de TPMs en el ámbito de programas computacionales es posible datarlo tan pronto como el mercado del software presentó su maduración total a principios de la década de 1980 (Smith, 2017)[68] y de ahí ha ido concurriendo con el desarrollo de los medios jurídicos de protección[69]. Aún con una tutela jurídica existente, los desarrolladores no han abandonado el desarrollo de más y mejores TPMs o empujar por mejor tutela jurídica, fundamentalmente porque uno u otro presentan límites inherentes. Sobre las TPMs en sí, se dice que existe una verdadera carrera tecnológica entre los desarrolladores de contenido digital sujeto a TPM y aquellos que desarrollan mecanismos para eludir dichos TPMs tornándolos un medio ineficaz (R. Kerr, Maurushat, & Christian, 2005)[70]. Es posible denominar a estas prácticas de elusión, en sus más diversas formas y expresiones, con el nombre de *circumventions*.

A esta altura, los TPMs no son un fenómeno realmente nuevo o extraño con respecto de ciertos ordenamientos jurídicos. En el desarrollo de las TPMs, y sin perjuicio de que muchos de estas medidas son en realidad programas susceptibles de protección jurídica directa por la vía de derecho de autor (ej. solo el desarrollador de la TPM podrá distribuirla), también existe un reconocimiento legal en ciertos ordenamientos, en la forma de protección especial, aunque ligada al ámbito del derecho de autor, en una suerte de suplementación, y cuyo objeto es la prohibición de actos de elusión de las medidas tecnológicas de que se valga un desarrollador (*anti-circumvention*). Entre los marcos legales más insignes a este respecto encontramos la polémica *Digital Millennium Copyright Act* de los Estados Unidos que data de 1998 (DMCA).

---

[67] Ob. cit. supra nota [65].

[68] Smith, E. (2017). *The Incredibly Technical History of Digital Rights Management*. Obtenido de *Vice Media Group:* https://www.vice.com/en_us/article/evbgkn/the-incredibly-technical-history-of-digital-rights-management

[69]Entre los medios tecnológicos más usuales en los programas están: los medios de validación o verificación de adquisición y cumplimiento de otras obligaciones por parte del usuario (ej. *product keys*); gestión y administración continua de los intereses del desarrollador a través de servidores (ej. *license servers*, autenticación en línea), uso de criptografía para verificar o restringir el acceso al programa o sus funcionalidades así como el uso de otro tipo de técnicas para resistir o repeler eventuales ataques de seguridad, modificaciones del programa, o su simple estudio (ej. ofuscación del código fuente), y la implementación de uno o más TPMs con hardware (ej. *dongles*).

[70] Ob. cit. supra nota [65].

De esta manera, si los TPMs buscan hacer efectivos los intereses de los desarrolladores, algunos sistemas jurídicos buscan a su turno dotar de reconocimiento y efectividad a los mismos TPMs, comprendiendo también debidos límites, esta vez jurídicos y no tecnológicos, de estos. Esto último es importante, y en ese sentido, ya se comprenden limitantes en los mismos sistemas anti-elusión (ej. Sección 1201 (f) de la DMCA), saltando a la vista aspectos relacionados con la libre competencia[71], derecho del consumidor o del consumo, y otros temas de derecho de propiedad intelectual (Välimäki & Oksanen, 2006)[72]. Valimaki y Oksanen (2006)[73] grafican la necesidad de tener a la vista los diversos intereses en juego, mostrando como el desarrollo de TPMs pueden servir para limitar u obstar injustificadamente a la interoperabilidad, principio rector en el ámbito de los sistemas computacionales. En nuestro país, la autora Javiera Plana (2016)[74] asoma que la protección jurídica de las TPMs, como verdadera suplementación a los derechos de autor, han conllevado, al menos en los sistemas ejemplares actuales (ej. DMCA), a un *desequilibrio neto* en perjuicio de los derechos de los usuarios.

Respecto de nuestro país, el estado del tratamiento jurídico de las TPMs es poco claro en el mejor de los casos, y completamente incierto en el peor. Las fuentes posibles de protección, además de la protección atingente que pueda gozar una TPM por ser un programa, siempre por una vía relacionada al derecho de autor, están en el Tratado de la Organización Mundial de la Propiedad Intelectual de 1996[75], ratificado por nuestro país y efectivo desde 2002, y el Tratado de Libre Comercio entre Chile y Estados Unidos ("TLC 2004"), vigente desde el año 2004 (Lara & Vera, 2014)[76]. Extrañamente, el marco legal más relevante al respecto de las TPMs y aplicable a nuestro país es el TLC 2004 referido. Con todo, la relevancia se reduce a un compromiso internacional que, eventualmente, desembocaría en una ley. De esta manera, a falta de ley vigente expresa, debemos decir que los posibles intereses jurídicos presentes en las TPMs se protegen según las reglas generales, sin protección especializada. En la clásica analogía del baúl (obra protegida), candado (TPM) y la llave (solución o medida elusiva de la TPM), que

[71] *Lexmark International, Inc. V. Static Control Components, Inc.* (6th Circ., 2014). Véase especialmente la opinión del juez Gilbert S. Merritt.

[72] Välimäki, M., & Oksanen, V. (2006). *DRM Interoperability and Intellectual Property Policy in Europe. European Intellectual Property Review (Vol. 26, No. 11)*, 562-568.

[73] Ibid.

[74] Ob. cit. infra nota [198].

[75] Dicho tratado exige a los estados parte, en su artículo II: "Protección jurídica adecuada y recursos jurídicos efectivos contra la acción de eludir las medidas tecnológicas efectivas".

[76] Lara, J. C., & Vera, F. (2014). *Medidas tecnológicas de protección de propiedad intelectual: desafíos regulatorios en Chile. ONG Derechos Digitales (Policy Paper No. 01)*, pp. 3-4; 7-10.

tan razonable es sostener que la "llave", autónomamente considerada (ej. se comercializa una medida elusiva o solución de TPM de manera autónoma), es una obra derivada, vale decir, un producto autónomo, pero *creativamente* relacionado al "candado". Luego, considérese un caso concreto en que se comercialice el mismo *baúl,* pero ahora sin candado (ej. versiones libres de TPMs). En el primer caso, a efectos de nuestro país, parece más dable valerse de una acción de las que contempla la Ley 20.169 de Competencia Desleal, y sin perjuicio de la posible infracción de las marcas comerciales. En el segundo caso, se puede estimar aplicable la LPICh. Por último, las operaciones que puede hacer un usuario de un programa destinadas a inutilizar una TPM de acceso o uso (principalmente la modificación del programa), en un contexto puramente personal (léase, para usarlo con su computador) parecen, como se verá en el Capítulo 7, excluirse del derecho de autor, nada obstando a que puedan incorporarse y perseguirse de manera y con efectos contractuales (ej. se prohíbe expresamente la ingeniería inversa y la modificación del código al usuario; se prohíbe la manipulación de cualquier TPM incluida o relacionada al programa).

La definición propuesta de "TPM" prevista en este trabajo, ha cuidado de no identificar la razón que motiva su creación y disposición efectivamente utilizada por el desarrollador como un derecho de propiedad intelectual o derecho alguno, utilizándose el término *interés*. La razón apunta a que los TPMs son también ampliamente utilizados para proteger intereses totalmente fuera de aquellos que disponen o se relacionan con los sistemas de propiedad intelectual (Välimäki & Oksanen, 2006)[77]. En general, habida cuenta que el ordenamiento jurídico nacional no ha recibido al fenómeno de las TPMs con normas especiales, pero constando que en los sistemas comparados actuales se relaciona la protección jurídica especializada de las TPMs con el derecho de autor, es que se estima como suficiente el panorama propuesto en este acápite.

3.2. Medios Jurídicos de Protección.

Según C. Phillips (1992)[78] y Samuelson et al (1994)[79] tecnologías como los programas necesitan protección, al menos legal, principalmente por existir una gran disparidad entre los costos incurridos en su desarrollo y el extremadamente bajo costo de copiarlo. Es interesante que este argumento, ya sostenido en la década de 1990, solo puede verse más y más reafirmado conforme han pasado los años.

[77] Ob. cit. supra nota [72].
[78] Ob. cit. supra nota [51].
[79] Samuelson, P., Davis, R., Kapor, M. D., & Reichman, J. (1994). *A Manifesto Concerning The Legal Protection of Computer Programs. Columbia Law Review (Vol. 94, No. 8),* pp. 2332-2342.

Podemos agregar a esta urgencia, el hecho de que la industria del software se caracteriza por ser impredecible, concentrada y altamente competitiva (Jullien & Sand-Zantman, 2016)[80], un ambiente especialmente hostil para los desarrolladores y otros entes activos (ej. inversores), por lo que ámbitos de protección jurídica de los programas serán imprescindibles para que no se desincentive la entrada a un mercado de importancia creciente.

Históricamente, la protección jurídica principal y más distintiva de los programas comenzó mediante la utilización de *contratos* y *secretos empresariales* (Hollaar, 2002) [81] (Samuelson, Davis, Kapor, & Reichman, 1994)[82], y se vio suplementada posteriormente con la entrada generalizada de *sistemas de derechos de autor y patentes*[83]. En ese sentido, gran parte de la protección jurídica de los programas termina siendo un fenómeno multidisciplinario de los distintos subsistemas que componen la *propiedad intelectua*l (Menell, Lemley, & Merges, 2019)[84].

Los contratos principales y distintivos relacionados a los programas son el contrato de licencia (cuyo objeto es la información, o más propiamente la divulgación de esta) y cuya expresión más común, desde la fase de masividad de la industria, es la contratación en forma de adhesión (Samuelson, Davis, Kapor, & Reichman, 1994)[85]. Tienen relevancia los distintos contratos de servicios de programación (cuyo objeto es un servicio relacionado al desarrollo de un programa). Otros contratos relacionados a los programas son los contratos de acceso y de uso de sitios web y que se relacionan principalmente al SaaS.

Un secreto comercial o empresarial refiere a información de distinta clase, efectivamente mantenida como confidencial, que goza de valor económico, y cuya adquisición, uso o divulgación se encuentra por estas razones limitada o restringida en concordancia, por lo general, a un sistema de propiedad

---

[80] Jullien, B., & Sand-Zantman, W. (2016). *Network Effects. Institut D'Economie Industrielle (No. 27)*, pp. 3-7; 11-12; 19-23.
[81] Hollaar, L. A. (2002). *Legal Protection of Digital Information*. Obtenido de *Digital Law Online: Software Copyright:* http://digital-law-online.info/lpdi1.0/treatise17.html
[82] Ob. cit. supra nota [51] pp. 2371-2378.
[83] Se hace hincapié que esta idea apunta a la protección jurídica más distintiva y generalizada. No hace falta hacer hincapié en la procedencia de marcas comerciales a propósito de un programa o las normas de competencia desleal cuando corresponda. También debe decirse que es posible datar patentes relacionadas a un programa ya del año 1968 (un algoritmo presente en el programa *AutoFlow*) y los primeros programas registrados en una agencia pública de derechos de autor datan ya de 1961. Con todo, al año 60, el impulso de la propiedad intelectual en los programas como medio jurídico de protección, aún en países líderes en la industria y sin perjuicio de los secretos comerciales, no puede catalogarse como generalizada.
[84] Menell, P. S., Lemley, M. A., & Merges, R. P. (2019). *Intellectual Property in the New Technological Age: 2019.* Clause 8 Publishing, p. 35.
[85] Ob. cit. supra nota [51].

intelectual, aunque es posible trazar el carácter híbrido de los sistemas de secretos empresariales (Menell, Lemley, & Merges, 2019)[86]. Por referir a una gran diversidad de información, un secreto empresarial sobre un programa puede traducirse en una igual de diversa cantidad de supuestos: los algoritmos o funciones del programa; el código fuente del programa; el código (fuente u objeto) de versiones antecesoras, menos estables o de prueba (versiones "alfa" y "beta"); la estructura del código fuente, y cierta documentación relacionada.

La protección de los programas por la vía de patentes, con objeto principal en las funciones del programa (Hollaar, 2002)[87], presenta sus primeros precedentes de *posibilidad de admisión* recién al año 1981 en Estados Unidos, cuna y actor principal en la innovación general de la industria. La protección de programas por la vía de patentes con respecto a los derechos de autor, satisface tibiamente los umbrales mínimos de claridad respecto de los requisitos particulares para su procedencia, no cuenta con la armonización internacional, impone requisitos formales mucho más rigurosos y con un alcance más acotado que el derecho de autor. Autores como Pianon (2004)[88], Posner (2012)[89] y Becker (2013)[90], detectan también que las patentes, por las particularidades de la industria de software (ej. FOSS, costos de copia), son medios jurídicos quizá contraproducentes.

[86] Ob. cit. supra nota [84] pp. 43-47.
[87] Ob. cit. supra nota [81].
[88] Pianon, A. (2004). *Trade Secret Vs. Open Source: And the Winner Is. Erasmus Law and Economics Review 1*, 47-75.
[89] Posner, R. A. (2012). *Do patent and copyright law restrict competition and creativity excessively?* Obtenido de *The Becker-Posner Blog*: https://www.becker-posner-blog.com/2012/09/do-patent-and-copyright-law-restrict-competition-and-creativity-excessively-posner.html
[90] Becker, G. (2013). *On Reforming the Patent System.* Obtenido de *The Becker-Posner Blog*: https://www.becker-posner-blog.com/2013/07/on-reforming-the-patent-system-becker.html

4. Naturaleza Jurídica de los Programas y el Concepto de Licencia.

Tal como detecta Sono (2008)[91], las distintas operaciones y actividades en la industria del software son complejas, pudiéndose distinguir efectivamente tres clases de actos u operaciones de relevancia jurídica diferenciada, aunque ciertamente interrelacionadas. En ese sentido, cuando alguna persona habla o alude a un programa a secas ("comprar software"), puede estar refiriéndose a tres fenómenos con efecto y relevancia jurídica diferenciada, esto sin perjuicio de la utilización, más incidental, del concepto de software a propósito de servicios:

a) Los *derechos de propiedad intelectual* relacionados al programa. Si el titular estima por *intercambiar* tales derechos, usualmente lo hará mediante la autorización o "licencia" de los mismos, conservando la titularidad en todo momento. No es posible identificar a un programa *como* un derecho de propiedad intelectual. Ciertos derechos de propiedad intelectual *pueden nacer* a propósito de un programa. Los principales implicadas son patentes, secretos comerciales, derechos de autor, y ciertamente marcas comerciales.

b) El *medio tangible* en que se encuentra contenido el programa. Por la interdependencia hardware y software es un aspecto ineludible y que refiere principalmente a dispositivos de memoria y almacenamiento, sea internos (ej. disco duro, RAM) o externos (ej. CD-ROM). Se trata de un aspecto felizmente abordado por el grueso de los ordenamientos jurídicos existentes a través de sus respectivos sistemas de propiedad civil (ej. Libro Segundo del Código Civil), al estar en presencia de verdaderas cosas corporales por lo general muebles. Ventas, arriendos, donaciones, son algunos de los actos jurídicos de relevancia. Sin perjuicio de lo anterior, este es también el aspecto menos importante en la industria de software, ya que el estado actual del intercambio de programas, software e información en general ha seguido una clara tendencia a simplificar y prescindir de este ámbito hasta cuanto sea posible.

c) La información o simplemente el *software*. En general, es posible determinar dos puntos de vista relativos al tratamiento jurídico de la información: comprenderla así y sin más, o bien, comprender, legal o jurisprudencialmente, una categoría nueva de cosas, "cosas digitales". Sobre la segunda vía,

[91] Sono, H. (2008). *The Applicability and Non-Applicability of the CISG to Software Transactions. C. B. Andersen, U. G. Schroeter (eds.). Sharing International Commercial Law across National Boundaries: Festschrift for Albert H. Kritzer Wildy, Simmonds & Hill Publishing, London, 2008,* pp. 512-526. Obtenido de *Institute of International Commercial Law: https://www.cisg.law.pace.edu/cisg/biblio/sono6.html*

diremos que, a efectos de la literalidad de nuestro sistema de propiedad civil, no existen tal comprensión. Según Jijena (1996)[92], la teoría de bienes implícita en los arts. 565, 577, 578 y 584, entre otros, todos del Código Civil, la información, independientemente de su soporte o formato, no es ni una cosa corporal ni incorporal. Es posible decir, al tenor de las mismas disposiciones citadas, que la contratación de software produce cosas incorporales (los derechos personales), pero malamente se puede decir que el software, en sí, es una cosa incorporal y, salvo que un tribunal nacional decida innovar y agregar categorías nuevas de cosas, tampoco se puede decir que sea una cosa digital, esto al menos con pretensión conceptual jurídica. Han sido fundamentalmente los tribunales europeos los encargados de visitar esta teoría de cosas digitales en los programas, aún a falta de regla legal clara que así lo admita en el propio sistema interno, basándose principalmente en que el software no contenido en hardware aparte (ej. software descargado) debe tener idéntico tratamiento legal al software si contenido en hardware aparte (ej. software en CD-ROM)[93] (Hojnik, 2017)[94].

Por otra parte, la primera vía interpretativa acepta que la información no es una *cosa* en sentido jurídico. Tampoco parece dable sostener que la información sea, en sí misma, un *hecho,* aunque esto no obsta a que un contrato u otra manifestación de voluntad, disponga hechos *a propósito* de la información *(ej. se prohíbe realizar una conducta con respecto a un programa)*. Esta es la única interpretación que estimada como consistente a la teoría del Código Civil y acorde con lo dicho en capítulos anteriores y la necesaria distinción tripartita que da apertura a este capítulo, y que permite concluir que no es posible hablar realmente de la naturaleza jurídica de la información, especialmente como una cosa corporal o incorporal. Que la información no sea una cosa implica que su intercambio, basado ciertamente en contratos, no está dado por ventas, arriendos, donaciones, u otros actos típicos que aplicamos al dominio, posesión o tenencia de *cosas*, por ser situaciones jurídicas solo aplicables correctamente al ámbito tangible y no a la información. Es posible utilizar dichas categorías reservadas a auténticas cosas por una vía analógica, como parece realizarse en la jurisprudencia norteamericana que luego se revisará.

---

[92] Jijena, R. (1996). *Copia ilegal de software y virus informáticos: un maridaje ilícito. Revista de Derecho de la Universidad Católica de Valparaíso (No. 17)*, pp. 313-330.

[93] El caso paradigmático en que se haría *necesario* tal plano de igualdad es el *agotamiento del derecho de autor de distribución*. Véase por ej. Hojnik (2017, ob. cit. infra: 74-78).

[94] Hojnik, J. (2017). *Technology neutral EU law: Digital goods within the tradicional goods/services distinction. Internation Journal of Law and Information Technology (Vol. 25, Is. 1)*, pp. 63-84.

Que la información no tenga naturaleza jurídica alguna no obsta a que el sistema de contratos sea un vehículo jurídico satisfactorio y suficientemente seguro, al que se le incorporan los sistemas de propiedad intelectual cuando corresponda. Tal vez lo único que debemos conceder bajo esta interpretación es que el objeto de estos contratos será el de *divulgar o difundir* la información de que consta, por ejemplo y ciertamente, un programa, algo que reconduce a una obligación de hacer al tenor del art. 1461 del Código Civil. El sistema de contratos se ha utilizado satisfactoriamente en la industria de software, a propósito de la información o servicios relacionados, por lo menos, desde la entrada del primer programa "vendido" (nótese la impropiedad) de manera separada al hardware (*AutoFlow* en 1965[96]).

En la jurisprudencia norteamericana del caso *Vernor v. Autodesk, Inc.* (WD Wash., 2008, "Vernor"), donde yace la filosofía de que "es el código del programa y sus derechos asociados lo que es valorable en contraste a los asequibles discos en que el código se encuentra almacenado". En este caso, el tribunal entra de lleno a abordar las situaciones jurídicas que una persona puede tener con respecto a la información, distinguiéndose "dueños" y "licenciatarios"[97] Según el tribunal de *Vernor*, una persona es comprendida como licenciataria, en contraste a verdadero dueño, cuando: a) se especifica que el acto antecedente se trata de un contrato de *licencia*; b) restringe significativamente la facultad del usuario de transferir el programa; c) impone notables restricciones de uso. Debe advertirse que por el concepto de "restricciones de uso", el tribunal no se refería a los usos propios subsumibles como derechos de propiedad intelectual, especialmente derecho de autor, vale decir, distribución, reproducción, comunicación y ejecución pública y adaptación, sino a la plana y simple ejecución del programa[98]. Es

[96] Es común que se cite a la empresa *Applied Data Research (ADR)* a propósito del programa *Autoflow* (1965) como pionero en la venta de software predefinido o productos de software necesariamente desvinculados de cualquier tipo de hardware (*independant software vendor*). En efecto, la *FreeBSD Foundation* sostiene que: "en el año 1965 *ADR* desarrolló el primer producto de software licenciado de manera independiente a una compañía de hardware (…) en 1968 ADR patentó su programa". La traducción es propia. Debe advertirse que en este contexto, "patentar un programa", aludía a ciertos procesos presentes en el programa en cuestión (un algoritmo para ordenar datos) y al código en sí. Nótese como el programa ya se "licenciaba", antes siquiera de estar afecto a derechos de propiedad intelectual (en este caso patentes y luego, posteriormente, derechos de autor) ¿Cuál fue entonces el *objeto jurídico* de dichas "licencias"?

Citas obtenidas de Bruce Montague (1995-2020). *Why you sould use a BSD style license for your Open Source Project*. Disponible en línea: https://svnweb.freebsd.org/doc?view=revision&revision=53942, consultada el 20 de mayo de 2020.

[97] Las situaciones jurídicas con respecto a la información también se abordan en *MAI Systems Corp. v. Peak Computer, Inc* (9th Circ., 1993).

[98] Se hace hincapié que, al menos en la parte analizada, no se discuten posibles patentes asociadas al programa. Si el programa implicado en el caso *Vernor* o de cualquier otro caso tuviera una patente asociada, entonces la literal ejecución del programa estaría ciertamente relacionado a derechos de propiedad intelectual. En cualquier caso, las licencias de software usualmente permiten, prácticamente como primera cláusula, la ejecución del programa, situación que puede llevar

patente que el concepto de *dueño* es utilizado por vía analógica y no en su concepto legal expreso, primero porque el concepto legalmente definido de dueño solo se aplica a cosas; segundo, porque "licenciatario" no es una situación jurídica civil alternativa a la de dueño, poseedor, o tenedor (sea en el sistema estadounidense o el nuestro), sin perjuicio que se reconoce en distintas sedes y subsistemas de la propiedad intelectual, pero que, por lo anteriormente dicho, no aplican al análisis; y tercero, porque el tribunal se hallaba en cierta medida forzado a utilizar "dueño" por la disposición legal que era objeto de análisis (Sección 107 (a) (1) del 17 U.S. Code). Aplicando dicho requisito en el caso *Blizzard v. MDY* (9th Circ., 2010, "Blizzard"), el tribunal admite que de entre estas restricciones que implican licenciamiento y no otorgamiento de propiedad (diríamos una "venta") está la imposición de restricciones para transferir el programa mismo (una suerte de "re-divulgación" de la información de software) y las copias tangibles que se puedan haber entregado o vendido. En definitiva, si alguien obtiene de otro un determinado bloque de información, independientemente de si está contenido en uno u otro medio, y el marco que regula la utilización de dicho bloque dispone que tal utilización es, en general, precaria, esencialmente temporal o muy restringida, aquella persona que recibió dicha información es, legalmente, un licenciatario.

La noción de licencia, licenciante y licenciatario en la industria debe datarse antes de que los sistemas de propiedad intelectual fueran utilizados de manera extendida por los desarrolladores a partir de la década de 1980 y apuntan a un contrato, el contrato que ya identificamos como el acto jurídico presente en el intercambio de información. No es correcto definir una licencia de software como un acto por el cual se autorizan derechos de propiedad intelectual a otro y sin más. Para aumentar las confusiones, estos actos que autorizan el ejercicio de derechos de propiedad intelectual en favor de terceros también se denominan licencias (como a los que alude, por ejemplo, nuestra Ley 19.039 de Propiedad Industrial en sus arts. 51 y ss., art. 77, y el art. 108). Estas licencias de derechos tienen por objeto *derechos* cuyos supuestos, contenido y facultades supuestos se disponen por el sistema de propiedad intelectual respectivo. Hay auténtica licencia de derechos de propiedad intelectual, específicamente, y al menos, derechos de autor, en el ya mencionado FOSS.

---

a comprender, de manera razonable, una licencia de patentes implícita en caso de que no se hubiese otorgado licencia expresa anterior.

En una licencia de software, comprendida como un acto jurídico cuyo objeto es la información, y más propiamente, la divulgación de esta, el centro mercantil u "objeto económico del contrato", es el software como información y no necesariamente los derechos implicados, aunque es usual a que se aluda a estos[99]. Pero la determinación del objeto jurídico es ambivalente, requiriendo examinar celosamente las partes de dicho acto jurídico. Esta situación de posible ambivalencia del concepto de "licencia", que parece configurar un acto complejo de partes con diversa naturaleza jurídica, se ha detectado de manera implícita en la jurisprudencia norteamericana a partir del caso *Sun Microsystems, Inc. v. Microsoft Corp.* (9th Circ., 1999). Aquí se comprende que el contenido de una licencia incluye aspectos de derechos de autor u otros derechos de propiedad intelectual, contenido este último denominado como "license conditions", y un contenido extraño a tal ámbito de naturaleza meramente contractual, bajo la denominación de "contractual covenant". Explicándose los criterios para distinguir entre uno y otro, en el caso *Blizzard*, el tribunal estima que el criterio decidor apunta a si *el acto* analizado en cuestión, según las cláusulas particulares objeto de la litis, es posible de subsumir o no como un derecho de propiedad intelectual. A efectos de los posibles derechos de autor implicados, esto implica algo quizás simple: el acto se encuadra efectivamente como una reproducción, adaptación, distribución o comunicación pública. Si el acto no puede encuadrarse como alguna de esta hipótesis, puede decirse que tal acto produce derechos personales y obligaciones de fuente contractual, aunque debe recordarse que un programa puede tener otra clase derechos de propiedad intelectual (ej. patentes sobre algún método presente en el programa) que deben unirse al análisis presentado. Nuevamente, y en todo caso, se reafirma la necesidad de separar las distintas acepciones de "programa" en el tráfico comercial.

En definitiva, si ha de aportarse un concepto de licencia de software que haga justicia a las transacciones que día a día ocurren a propósito de la información y que también aborde todos los ámbitos de ambivalencia conceptual (ej. licencia de derechos v/s licencia del programa, covenants v/s conditions, ámbito tangible v/s ámbito intangible, etcétera), la falta de una naturaleza jurídica propia de la

[99] Cuando el FOSS alude al derecho de autor, generalmente es a efectos de dar las autorizaciones de uso atingentes y correspondientes (ej. licencia de adaptación y distribución). Cuando el software propietario alude al derecho de autor, generalmente es para únicamente reafirmar la titularidad (ej. prohibiendo la modificación y la distribución). Solo respecto del primero, y en este ejemplo particular, podríamos decir que se presenta una auténtica *licencia de derechos de autor*. Con todo, ambos actos son *licencias de software*, porque en ambos el *objeto económico* es la información, y especialmente, distintos actos relacionados a esta. El *objeto jurídico* en las licencias serán así también actos, a falta de categoría legal de cosas digitales en el sistema particular.

información en nuestro sistema y la historia y caracteres propios de la industria (ej. *freemium*), diremos que se trata de un *acto jurídico complejo* que tiene por objeto la divulgación de la información de que trata el programa[100], sea que exista o no un precio por la divulgación en sí[101], y que dispone un marco detallado que regula la utilización de la información así difundida, luego, según la naturaleza de los actos específicos dispuestos en dicho marco[102], es que se hace posible determinar las normas jurídicas concretas aplicables y en definitiva determinar la naturaleza jurídica de cada parte, componente o elemento de este acto jurídico complejo, la cual puede tener, generalmente, fuente contractual o fuente en algún subsistema de propiedad intelectual, como puede ser el sistema de derecho de autor[103].

[100] En nuestro sistema no es posible decir, al menos jurídicamente, que una licencia de software es "*un contrato cuyo objeto es el programa"*. El objeto jurídico de los contratos se dispone en los arts. 1460 y 1461 del Código Civil (cosa o hecho humano). Definir un programa como un "acto" es, sorpresivamente, más acertado (al no subsumirse como cosa), pero es quizás incoherente para la forma ordinaria de emplear los términos. Convenientemente, los derechos de propiedad intelectual refieren no a cosas sino a actos (ej. el derecho de autor habla de reproducción, el derecho de patente habla de explotación comercial). De esta manera, se llega así a la conclusión de que el objeto jurídico en las licencias de software son actos, siendo el primero de estos, el de divulgación.

[101] Esto es lo que algunos denominan como "entrega" del programa. Nótese que en el caso del software de retail, la entrega del programa (información) concomita con la entrega del contenedor físico (hardware). En software descargado no hay entrega de aspecto físico alguno, pero sí que es posible hablar de que se ha "entregado" la información. En el SaaS no hay entrega de información alguna, y ciertamente tampoco de hardware adicional, y en realidad, simplemente se prestan servicios.

[102] Esto alude, fundamentalmente, a los actos que permite realizar la licencia (ej. ejecutar públicamente el programa v/s ejecutarlo personalmente). Algunos tribunales, como el *9th Circ.* en el caso *Blizzard* y el *Fed. Cir.* en el caso *Jacobsen* (ambos ya mencionados), incluyen como elemento adicional de análisis, la forma en que se halla redactada y formulada (ej. no es lo mismo contraer la obligación o la promesa de autorizar uno u otro acto, que disponer que un acto está desde ya autorizado).

[103] Se advierte que esta definición excluye, con justa razón, el caso del software prestado como servicio (no se obtiene información alguna, el usuario accede a funciones o servicios de cómputo mediante internet).

5. Derecho de Autor Chileno en los Programas: Generalidades del Derecho de Autor.

Según la Organización Mundial de la Propiedad Intelectual (en adelante "OMPI" o "WIPO" por sus siglas en inglés), en las décadas de 1970 y 1980 existió discusión interna a propósito de los medios jurídicos a utilizar para proteger a los programas. Se trata de un período en que la importancia de la industria iniciaba su irremediable maduración y donde la masividad ya se asomaba una consolidación, por lo que tales discusiones ocurrían también en los países donde la industria era relevante.

Los candidatos barajados por la OMPI para dar tutela satisfactoria a los programas eran el derecho de autor, derechos de patente, considerarlos como secretos empresariales, o derechamente concebir un sistema especializado inédito o *sui generis,* como el que existe, en ciertos países, con respecto a la topografía de circuitos integrados (aplicable a nuestro país) y el *contenido* de bases de datos con independencia de su estructura creativa (si es que siquiera existe). Según la misma OMPI, abandonado el proyecto de protección sui generis, es que comenzó a prevalecer la idea de aplicar el sistema de derecho de autor para dar protección. En una reunión conjunta con la UNESCO, a eso de febrero de 1985, se llegó al reconocimiento generalizado por los estados participantes de que los programas son sino obras susceptibles de ser amparadas por derecho de autor, aunque observando ámbitos de debida adaptación debido a las particularidades de la industria.

En realidad, a 1985 había ya 5 países que dotaban de protección a los programas por la vía de derecho de autor. Más aún, según la *Copyright Office* de los Estados Unidos (uno de esos 5 países aludidos) y las bases de datos con las que cuenta la referida, el primer programa efectiva y válidamente registrado data del 30 de noviembre de 1961 (Hollaar, 2002)[104][105]. Para la *Copyright Office*, pese a que los programas no se encontraban inequívocamente comprendidos como obras protegidas en el 17 U.S. Code y que podían existir objeciones para dotarles de protección[106], un programa simplemente se asimilaba a una especie de libro práctico estilo "hágalo usted mismo", esto último porque se comprendía atinadamente que

[104] Hollaar, L. A. (2002). *Legal Protection of Digital Information*. Obtenido de *Digital Law Online: Software Copyright:* http://digital-law-online.info/lpdi1.0/treatise17.html

[105]La fecha en cuestión es un aspecto no menor. Hollaar no lo precisa, pero debe advertirse que en dicho país el registro de la obra era requisito para el otorgamiento de protección de obras no publicadas hasta 1976.

[106]Siguiendo a Hollaar (2002, ob cit. supra nota [104]), quien probablemente tuvo a la vista el informe de la *National Commission on New Technology Uses of Copyrighted Works* a que se aludirá más adelante, una de las objeciones era un precedente de la Corte Suprema en el caso *White-Smith Music v. Apollo* (1908) que trataba sobre el estatus jurídico en el derecho de autor de las tarjetas perforadas (que sería una especie de programa primitivo) utilizadas en pianolas (que sería la máquina que instruir).

debía utilizarse con un computador físico, y que en suma se encontraba protegido tal como un libro, agregándose luego algunas pautas para facilitar la admisibilidad de los registros[107]. Aún con estos pioneros en el ámbito de protección por la vía de derecho de autor, Estados Unidos igualmente siguió la transición general de los países hacia una protección inequívoca, y muy especialmente, con ámbitos de adaptación a las exigencias propias de los programas y la industria. Cobraron gran relevancia para esta transición las recomendaciones de la *National Commission on New Technological Uses of Copyrighted Works* (en adelante, se tenga a la vista la sigla "CONTU")[108], ya establecida al año 1974, fijadas en un reporte cuya relevancia histórica y doctrinaria es muy importante para determinar los ámbitos de adaptación que tuvo que padecer el derecho de autor para incluir a los programas en una forma coherente y efectiva.

En nuestro país, nuestro marco más general del derecho de autor, la Ley 17.336 de Propiedad Intelectual ("LPICh"), *inició* su transición hacia la protección inequívoca de los programas a partir de 1985 con la Ley 18.443, introduciéndose el concepto de "programa y soporte lógico de ordenador". Este concepto no fue introducido en clave de concepto legal expreso, sino a propósito del pago que debía hacerse para registrar dichas obras, en una suerte de reconocimiento implícito de la protección. Fue en 1990 cuando la transición se sella con la Ley 18.957, mediante un concepto legal de *programa computacional*, la consagración protección expresa de estos y normas especiales incluidas en la misma LPICh. Importante fue la modificación que introdujo la Ley 19.912 de 2003, al comprender el *código fuente* en el concepto legal de programa computacional.

Por esta importancia histórica, incontestabilidad y especialidad general del tratamiento que se encuentra consolidada en prácticamente todos los países del planeta, así como su superioridad protectora con respecto a los derechos personales sin efectos absolutos nacidos de un contrato, es que los sistemas de derecho de autor son un medio jurídico de protección particular e importante en los programas. Otro aspecto relevante en la aptitud protectora del derecho de autor en los programas, es el principio de protección automática que prescriben la mayoría de estos, situación que dota de aún mayor margen protector a una clase de obras que componen una industria impredecible, de rápido crecimiento, donde hay actores informales relevantes (ej. programadores independientes que licencian

[107]Circular 61 de la *Copyright Office* (1964): *Copyright Registration for computer Programs.*
[108]Se trata de un documento público de los Estados Unidos y de coautoría de expertos en propiedad intelectual de dicho país (1978). Disponible en línea: https://files.eric.ed.gov/fulltext/ED160122.pdf, consultada el 3 de marzo de 2020.

sus programas en repositorios web) y que contrasta con medios de protección más formales y restrictivos como el patentamiento.

5.1. Sistemas de Derecho de Autor.

Doctrinariamente, el derecho de autor es un subsistema inserto en un sistema legal más general denominado propiedad intelectual. La propiedad intelectual es un marco jurídico diferenciado del sistema de propiedad civil de cosas tangibles, cuyo objeto es la protección de esfuerzos intelectuales esencialmente intangibles (Menell, Lemley, & Merges, 2019)[109].

En el derecho de autor, el esfuerzo intelectual toma la forma de una obra literaria o artística, y la protección concreta se da mediante el otorgamiento de derechos exclusivos, temporales, de carácter erga omnes normalmente en favor del autor material de la obra y cuyo contenido fundamental apunta a explotar económicamente la obra. También se reconocen derechos extrapatrimoniales o morales que permiten a sus titulares contar con medios para preservar y proteger la paternidad e integridad de sus creaciones, y ámbitos de protección en favor de otros interesados que han contribuido a la difusión de una obra protegida o que han desarrollado o creado un trabajo que requiere cierta creatividad y habilidad intelectual y que se denominan como derechos conexos o *neighbouring rights* (OMPI, 2016)[110]. Es común sostener que el fundamento del derecho de autor es el progreso de las artes y el acervo creativo de la comunidad, balanceando los intereses económicos y morales del creador de la obra con el interés del público general a acceder a esta.

Las fuentes fundamentales del derecho de autor, alcanzan primero las normas legales internas, que aquí se denominan *sistemas de derecho de autor.* Estas normas jurídicas serán efectivamente expresiones de un sistema de derecho de autor cuando regulen, independientemente de su rango o técnica legal, algunos de los aspectos o elementos de la definición de apertura. En ese sentido, también pueden existir normas internas de carácter meramente reglamentario atingentes. Nuestro sistema de derecho de autor se identifica con la Ley 17.336 de Propiedad Intelectual (“LPICh”) y sus modificaciones: una ley especializada en la materia, relacionada sistemáticamente al art. 584 del Código Civil, y que data del año 1970, la cual, al menos jurídicamente, nace y se coordina con la exigencia constitucional del artículo

[109] Menell, P. S., Lemley, M. A., & Merges, R. P. (2019). *Intellectual Property in the New Technological Age: 2019.* Clause 8 Publishing, p. 16-22.
[110] Organización Mundial de la Propiedad Intelectual. (2016). *Understanding Copyright and Related Rights.* pp. 3-19.

19 N°25 de nuestra Constitución, que dispone la libre difusión de las artes y ciencias y del derecho de autor que se hace así derecho fundamental y que cuenta con importantes garantías (ej. art. 19 N°26, recurso de protección). Eso sí, adviértase la impropiedad doctrinaria de su denominación.

Aun teniendo en cuenta que los sistemas de derecho de autor se rigen bajo el principio de territorialidad, producto de los esfuerzos de coordinación entre estados y la importancia generalizada que la promoción del progreso intelectual humano posee para estos, los tratados internacionales atingentes son también fuente del derecho importante. En ese sentido cabe mencionar la Convención de Berna para la Protección de las Obras Literarias y Artísticas ("Convención de Berna"), datada en 1886 y que hoy se encuentra ratificada por 177 estados; y el tratado internacional que establece la OMPI que se encuentra vigente en Chile desde junio de 1975. Además de estos dos marcos generales, a efectos de la temática de este ensayo y capítulo particular, conviene traer a colación el Tratado de la OMPI de 1996 sobre Derecho de Autor ("Tratado OMPI 1996") vigente en nuestro país desde marzo de 2002 y la Parte II Sección 1 del Acuerdo sobre los aspectos de los Derechos de Propiedad Intelectual relacionados con el Comercio de 1994 y vigente en nuestro país desde 1995.

5.2. Objeto: obras protegidas.

El objeto del derecho de autor es una obra literaria, en el sentido de escrita o artística. También es usual que los sistemas contengan un catálogo ejemplar, meramente enunciativo, de obras protegidas (ej. art. 3 de la LPICh en concordancia al art. 1 de esta).

Con todo, no toda obra del intelecto queda alcanzada por el derecho de autor. Para que una obra sea susceptible de ser protegida por esta vía, problema que aquí se pasa a denominar "elegibilidad" de la obra, en relación al concepto extranjero más propio de *eligibility*, esta debe reunir las siguientes características:

1. La obra debe estar fijada en un soporte material. Solo importa que la obra esté reproducida en un soporte suficiente a efectos de dotarla de permanencia, dando así la posibilidad de que sea percibida por los sentidos, sin importar si el medio o soporte es tanto o más efectivo a ese efecto. A este efecto debe comprenderse que creaciones humanas con existencia digital (como un programa) cumplen siempre este requisito, ya que el soporte no es otro que el computador físico. Diríamos que no existe el software en una burbuja aparte del hardware. Si bien en nuestro país no se reconoce expresamente

este requisito, la esencia de este apunta que se excluyen los pensamientos y las operaciones mentales no exteriorizadas, por más creativas que sean.

2. La obra debe ser creada por un ser humano. En ese sentido, no existen derechos sin un titular, aunque, si bien el autor material de todo esfuerzo intelectual humano será precisamente un humano, nada obsta a que el titular pueda ser una persona jurídica, aún e incluso, de manera originaria, tal como parece consagrar el art. 8 inciso 2° de la LPICh, precisamente a propósito de los programas.

3. La obra debe ser original. La originalidad en el ámbito del derecho autor refiere a que la expresión particular de la o las ideas de las que se vale el autor, sea única y valorable, en el sentido de que solo en estos casos existe efectivo esfuerzo intelectual. No es necesario que las ideas en sí sean originales. No se cumple este requisito, precisamente, cuando la creación es copia de una obra existente ya protegida.

Ciertas obras, no obstante ser *elegibles* según lo antedicho, se excluyen del ámbito de protección del derecho de autor, siendo el caso de las sentencias judiciales, dictámenes y reglamentos administrativos, entre otras (OMPI, 2016)[112]. Ciertos sistemas también han prescrito (en algún momento de su historia) la necesidad de requisitos adicionales a la simple elegibilidad, como la inscripción de la obra en algún registro público o su necesaria publicación.

El hecho de que no se requieran requisitos externos aparte de la simple *elegibilidad* a efectos de otorgar protección, se conoce como principio de protección automática. Esta circunstancia es extremadamente relevante al conferir y abordar la protección en obras que no tienden a publicarse de manera formal como el efectivo caso de los programas (Hollaar, 2002)[113]. No es contrario al principio de protección automática el hecho de que se prescriban requisitos con un alcance distinto al nacimiento mismo de los derechos de autor (ej. efectos probatorios). A efectos de nuestro sistema, nuestra LPICh y el Convenio de Berna, consagran la protección automática explícita e implícitamente, en sus respectivos artículos 1[114].

5.3. Derechos patrimoniales en general, titular y otras facultades.

[112] Ob. cit. supra nota [110].
[113] Ob. cit. supra nota [104].
[114]El Convenio de Berna dispone que: “Sin embargo, será materia de la legislación de los países contratantes la circunstancia de prescribir que las obras, en general o según tipo, no serán protegidas salvo que se encuentren fijadas en algún soporte material”. La traducción es propia. Luego, a contrario sensu, podríamos decir que los estados contratantes no pueden imponer restricciones sobre la elegibilidad de una obra para obtener protección que no se basan en la necesidad de fijeza.

El supuesto de hecho del derecho de autor es la creación de una obra elegible. Los derechos subjetivos otorgados por el derecho de autor implican que solo quien sea el titular puede ejercitar o realizar ciertos actos con respecto a la obra. En ese sentido, la característica esencial de los derechos de autor es su exclusividad.

Los derechos de autor patrimoniales, según la visión más canónica que delimita su contenido, son:

a) Reproducción de la obra, en clave de hacer copias de esta, mediante cualquier procedimiento y en cualquier soporte.

b) Distribución de la obra o de sus copias. Según la definición de la LPICh, la distribución consiste en "la puesta a disposición del público del original o copias tangibles de la obra mediante su venta o de cualquier otra forma de transferencia de la propiedad o posesión del original o de las copias"[115]. Se incluye en este concepto de distribución, el arrendamiento o *rental of copies*[116], que se explora por los sistemas de derecho de autor a propósito de ciertas especies de obras. A propósito de los límites del derecho de distribución en relación a la extensión territorialidad de los sistemas de derecho de autor, es que algunos de dichos sistemas consagran el derecho de controlar las importaciones (OMPI, 2016)[118].

e) Derecho de publicar la obra, y de comunicarla o ejecutarla públicamente. Estos derechos se relacionan con actos destinados a difundir la obra. En nuestro sistema, en base al art. 18 letra a) podemos determinar, con cierta base, que la publicación de la obra es una especie de comunicación pública. La LPICh no define el derecho de ejecución pública, pero si delinea sus contornos de manera implícita al tratar la definición de comunicación pública, por lo que bien puede ser otra especie de esta[119]. Si examinamos el concepto de la LPICh de distribución, podríamos incluso concluir que la distribución no es sino una *comunicación pública con copias tangibles*. Es en la definición legal de comunicación pública donde puede extraerse el alcance del concepto de "público" conviniendo también

[115]Véase el artículo 5 letra q) de la LPICh.
[116]La Directiva 2009/24/EC de la Unión Europea, sobre Protección Legal de los Programas Computacionales, en lo dispuesto en el artículo 4.1. letra c) parece, al igual que en este ensayo, contemplar el *rental right*, también como una especie de derecho de distribución.
[118] Ob. cit. supra nota [110].
[119]Piénsese en los ya muy conocidos *streamers* de contenido digital protegido, los cuales, sin perjuicio de hallarse físicamente en la comodidad de su hogar, transmiten contenido que efectivamente se radica en el lugar físico de los miles y millones de espectadores, todo al mismo tiempo. Esta circunstancia es simplemente uno de los tantos productos de las capacidades de procesamiento de información que tienen los sistemas computacionales (y las redes de estos) en la actualidad.

citar el art. 71 N de esta. En general, la noción de lo público debe relacionarse a la naturaleza patrimonial o económica propia de los mismos.

e) Derecho de adaptar, traducir o modificar la obra originaria. Las obras resultantes de esta clase de actos se denominan obras derivadas en la medida que sean eligibles de protección según los criterios ya sostenidos. Nuestra LPICh, en su art. 5 letra i, parece acorde a este comentario al prescribir que una obra derivada es tal en la medida que constituya "una creación autónoma".

Sobre estos derechos operan limitaciones y excepciones que restringen su alcance. En ese sentido, diversos sistemas distinguen entre usos libres y usos justos ("fair uses") que toman en cuenta la naturaleza y propósito del uso (ej. comercial o no comercial), la naturaleza o tipo de obra utilizada, la extensión de la obra efectivamente utilizada, y el posible efecto del uso en el valor comercial de la obra. Nuestra LPICh comprende esta idea bajo el concepto de *limitaciones y excepciones*, las cuales se disponen, en su grueso, en el Título III de la misma.

Los derechos anteriormente previstos con su contenido precisado por las limitantes que correspondan, son los derechos patrimoniales que corresponden otorgar ante el supuesto de una obra susceptible de protección. Cuando se aluda en este ensayo al concepto de "derecho de autor", se advierte que no se comprenderán ni los derechos conexos, ni los derechos extrapatrimoniales. Respecto de la exigibilidad de estos derechos de autor, vale decir, el catálogo de acciones, los sistemas de derecho de autor comprenden medidas especiales o adicionales, que se insertan en el sistema general de acciones. Según la OMPI, las medidas pueden clasificarse en provisionales (medidas cautelares), civiles, penales, en fronteras, y aquellas que se existen en circunstancias de abuso o elusión de TPMs. Existen países en donde las pretensiones jurídicas nacidas de los derechos de autor tienen atribuida una naturaleza especial, como el caso de Estados Unidos, donde se habla de *copyright infringenment* (con una teoría jurisprudencial que reconduce muchas veces a la legislación y jurisprudencia nacional en materia de civil extracontractual) y no de "acciones civiles o penales".

Respecto del problema de la titularidad de los derechos, el titular originario de los derechos de autor, por regla general, será el autor material de la obra. También es posible comprender excepciones a esta regla general donde, por diversas razones, la titularidad se radica en persona distinta. Claramente relacionadas a la titularidad son las situaciones de transferencia y transmisión de derechos que pueda comprenderse en el sistema, así como la extinción de los derechos, incluyéndose aquí la posibilidad de

renuncia. Encuadrando lo anterior, a modo ejemplar, nuestra LPICh consagra la regla general de la titularidad originaria del autor material en su art. 1 y las excepciones a la titularidad en el art. 8 inciso 2° y 5 letras b y c; la facultad de cesión de derechos patrimoniales que le asiste a todo titular en el art. 17, el concepto de titularidad secundaria en el art. 7 y la forma y requisitos de dicha cesión en el art. 73; y una especie de catálogo de supuestos de extinción de derechos de autor en el art. 11, incluyendo, al menos a priori, la procedencia expresa de la renuncia.

Además de la facultad de ceder sus derechos, el titular, sea originario o secundario, también cuenta con la facultad de autorizar o permitir la utilización de la obra en favor de un tercero, conservando en todo momento la titularidad. Con todo, si el permiso es otorgado de manera exclusiva, el titular pierde la facultad para conceder un permiso similar a otros. Estos permisos, también llamados licencias, sean exclusivos o no exclusivos, son supuestamente actos unilaterales del titular, vale decir, producen sus efectos con independencia de que el licenciatario acepte o no los términos. A esto se suma el hecho de que las licencias de derechos de autor son una categoría de actos jurídicos no contemplada por los sistemas civiles y que, por ende, se regula aplicando primero la ley especial correspondiente, aunque en nuestro caso, la LPICh parece asomar, en su art. 20, que una licencia debe necesariamente ser un contrato y que por ende produce efectos solo formándose el consentimiento entre el titular y el tercero interesado[120]. En ese sentido, la violación de los términos de la licencia sería, o estaría relacionado a, el derecho de contratos (incumplimiento), circunstancia que reporta varias dudas[121]. En cualquier caso, el alcance de los derechos licenciados debe entenderse restringido celosamente a sus términos, sea por comprenderse que los derechos de autor son exclusivos del titular o por disposición de nuestra propia LPICh. En ese sentido, el ya citado art. 20, en su inciso final, dispone que "a la persona autorizada no le serán reconocidos derechos mayores que aquellos que figuren en la autorización, salvo los inherentes (…) según su naturaleza".

[120] El art. 20 de la LPICh prescribe que: "se entiende, por autorización el permiso otorgado por el titular del derecho de autor, *en cualquier forma contractual*".

[121] Si un desarrollador de programas permite a un sujeto distribuir copias tangibles dentro de cierta parte del territorio, y el licenciatario realiza la distribución en territorio diverso, este último ¿Estaría infringiendo derechos de autor o una obligación contractual? En ese mismo sentido, si un programador otorga una licencia "irrevocable" para preparar obras derivadas en favor de un tercero, y decide, en cierto momento, revocarla ¿Cuál sería el o los efectos de dicha revocación?

6. Derecho de Autor Chileno en los Programas: Introducción

Las características generales del derecho de autor fueron esbozadas en el Capítulo 5. Ahora bien, la determinación general del alcance protector del derecho de autor específicamente abocado a los programas se articula sobre dicha base general, pero con importantes adaptaciones, sea en forma de normas especiales expresas o principios más difusos, pero en cualquier caso, modificaciones sustantivas relevantes y no meramente formales. En ese sentido se debe tener en cuenta que los sistemas de derecho de autor fueron concebidos, en un principio, para proteger obras artísticas y literarias más clásicas como novelas, poemas, obas cinematográficas, y no tan técnicas como es el caso de los programas.

La primera adaptación tiene relación con el carácter eminentemente funcional de los programas (instruir una máquina) y la segunda con la forma en que las transacciones que recaen sobre los programas se dan en la práctica (Samuelson, Davis, Kapor, & Reichman, 1994)[122]. Finalmente, se advierte que en este acápite se utiliza abundante jurisprudencia norteamericana, al punto que se estima como fundamental, principalmente porque el aludido país, foco de la industria de software (y de incontables litigios relacionados a esta), ha sido uno de los países que más ha debido probar la "elasticidad" o "acomodamiento", sea legal o jurisprudencial, de su sistema de derecho de autor al abordar a los programas. En ese sentido, debe comprenderse que, en lo que al grueso del software desarrollado en el mundo se refiere, los actores más relevantes, por lejos, son Estados Unidos y China. Por la territorialidad de los sistemas de derecho de autor, esta desmedida importancia de la autoría china y estadounidense de programas debe siempre tenerse en cuenta, aunque, hoy en día, el acceso a internet y la presencia de repositorios de programas, permite que otras partes del mundo, y sus respectivos sistemas de derecho de autor, cobren mayor relevancia. Esta mayor heterogeneidad en la autoría de programas también hará que los tratados internacionales atingentes cobren relevancia.

6.1. Concepto legal de programa e implicancias.

[122] Samuelson, P., Davis, R., Kapor, M. D., & Reichman, J. (1994). *A Manifesto Concerning The Legal Protection of Computer Programs. Columbia Law Review (Vol. 94, No. 8),* pp. 2315-2330; 2332-2342; 2371-2378.

El contenido de este acápite pasa a analizar de manera exhaustiva el alcance de este primer elemento presente en la protección de los programas por el derecho de autor: el recogimiento expreso e inequívoco de los programas como obras protegidas.

Nuestro sistema se adscribe a un modelo general de protección de programas estructurado a partir de un *recogimiento expreso* y sistemáticamente inequívoco[123] de los programas como obras susceptibles de protección y que data de 1990 con la entrada en vigencia de la Ley 18.957. Nuestro país incluye también una definición legal de "programa" y de "copia de programa". Es razonable presumir que nuestro país se nutrió ampliamente de los aportes de las reuniones de la OMPI del año 1985, así como de la experiencia de otros países que ya habían consagrado la protección expresa. En ese sentido, también cobraron efectiva relevancia ciertos instrumentos internacionales.

6.1.1. Tratados internacionales relevantes[125].

En este acápite se tratan dos instrumentos internacionales que vienen a suplementar a nuestra LPICh, en forma jurídicamente relevante, en cuanto a precisar el concepto legal de programa a la luz del derecho de autor. El Tratado de la OMPI de 1996 referido, fue ratificado por Chile desde 2001 y en vigencia desde marzo de 2002 se trata de un importante cuerpo legal que contextualiza y clarifica ciertos aspectos omitidos por la LPICh. Lo propio puede decirse del acuerdo sobre los ADPIC de la Organización Mundial del Comercio (OMC) dispuesto en 1994 y vinculante en Chile desde 1995, año en que nuestro país se hizo miembro de esta.

A) ARTÍCULO 2 DEL TRATADO DE LA OMPI DE 1996.

Esta norma sostiene que "el ámbito de protección del derecho de autor abarcará las expresiones pero no las ideas, métodos de operación o conceptos matemáticos en si".

Tal como se sostendrá más adelante, por ser los programas esencialmente algoritmos (pasos que arriban a una solución), esta norma se convierte en una disposición clarificadora necesaria, sin disposición análoga en nuestra LPICh, y de ahí su importancia. Los programas son el ámbito esencialmente funcional de un computador físico y en ese sentido, por ser el derecho de autor un ámbito preocupado por las

[123] A ese efecto, no considero "sistemáticamente inequívoco" el recogimiento que hizo la Ley 18.443 de 1985.

[125] Véase también: *Digital Ecosystem* (OMPI, 2007). Disponible en línea: https://www.wipo.int/edocs/mdocs/copyright/en/wipo_ip_cm_07/wipo_ip_cm_07_www_82573.pdf, consultada el 3 de marzo de 2020.

expresiones de las ideas más que las ideas en sí, al hacer referencia inequívoca de que también los métodos de operación y los conceptos matemáticos en sí no están protegidos, es que el alcance protector real del derecho de autor en los programas comienza a concretizarse, en el sentido de restringirse. Por de pronto, podemos decir que las "ideas" no protegibles presentes en una obra, al aplicarse conceptualmente a un programa, reconducen a las funciones del mismo.

Aún con el carácter bastante técnico y funcional de las ideas que subyacen en el desarrollo de programas, no hay razón para estimar que la creación de programas no conlleva un esfuerzo intelectual susceptible de protección, vale decir, en un programa concreto ciertamente habrá funciones no protegibles, pero no por eso será imposible encontrar expresión original si protegible. Este es el alcance de la expresión "en sí", vale decir, las funciones autónomamente consideradas escapan de los derechos de autor, pero, si ha habido expresión original al desarrollar el programa y sus funciones particulares, aún si su naturaleza funcional es ineludible, es que será posible dar protección. Esta circunstancia no es nada nueva para los sistemas de derecho de autor que consagran la protección de obras con carácter técnico o científico, como la protección expresa que hace nuestra LPICh sobre los "proyectos, bocetos y maquetas arquitectónicas". Piénsese también que tanto obras literarias altamente creativas como novelas de mundos ficticios y poemas llenos de metáforas son tan susceptibles de protección como un libro que explique fenómenos científicos o hechos históricos. Tampoco es posible negar que relacionado a un programa particular pueden concurrir obras más tradicionales del derecho de autor, como lo son las composiciones musicales e imágenes en un videojuego y los diseños de interfaces de usuarios (ventanas, íconos, bordes, etcétera).

Tal como observa Hollaar (2002)[126] la pregunta sobre *qué protege* el derecho de autor en un programa se halla simplemente *complicada* por su naturaleza funcional. ¿Es posible encontrar expresión original y libertad creativa en el desarrollo de programas? La respuesta debe ser afirmativa. No es posible aplicar la doctrina norteamericana de la fusión idea-expresión, al menos a priori, en un programa. Según este criterio o doctrina, nacido en la jurisprudencia norteamericana del caso *Baker v. Selden* (SCOTUS, 1879), si las formas de expresar una idea son limitadas, se produce una fusión entre la idea y la expresión, cuestión que acarrea que la expresión concreta escape del ámbito del derecho de autor. Esta doctrina

[126] Hollaar, L. A. (2002). *Legal Protection of Digital Information*. Obtenido de *Digital Law Online: Software Copyright:* http://digital-law-online.info/lpdi1.0/treatise17.html

alude a si puede decirse razonablemente que hubo (o no) libertad creativa en el desarrollo de una obra y sus partes.

Ya se había sostenido que el desarrollo de programas puede verse como el desarrollo de algoritmos: conjunto de pasos bien definidos encaminados a obtener un resultado. En realidad, para arribar al resultado, identificado como la operación que se busca que el computador haga, los programadores siguen las "secuencias" que estimen conducentes (The Linux Information Project, 2005)[128], existiendo distintos valores o principios de programación que el programador puede considerar y luego ponderar, muchas veces, según su propia personalidad (eficiencia, elegancia en la obtención de la solución, robustez, legibilidad del código, entre otros). Ninguno de estos principios son ámbitos de derecho de autor per se, pero grafican el abanico de posibilidades efectivamente creativas que existe en la programación. En ese exacto sentido, la CONTU en su aludido informe de 1978 ya expresaba y comprendía que la programación es, en sí, una forma de escritura humana que puede entenderse, de manera relativamente fácil, por otros humanos. Además, bajo la noción de diseño modular, como un programa puede comprenderse como varios componentes interrelacionados, la propia disposición de dichos componentes (concepto que a veces se denomina como la "estructura del programa", un ámbito relevante conforme a el Capítulo 6.2.2. de este trabajo) puede ser un actividad altamente creativa. Podemos así identificar los resultados buscados por el algoritmo con la funcionalidad o tarea que realiza el programa ("idea"), y el proceso o secuencias de pasos que elige el programador con la particular forma de expresión[129].

Por último, todo programa inicia con un código, un texto escrito en un lenguaje de programación y muchas veces con profusa documentación relacionada al diseño de dicho código y que persigue fines explicativos, hasta didácticos, y no utilitarios. Cobran especial relevancia, como forma de documentación, los comentarios al programa incluidos en el mismo código fuente que hace el desarrollador para explicarlo, estando de hecho destinados para ser leídos por una persona.

b) Art. 4 del Tratado OMPI de 1996 y el Art. 10 del ADPIC de la OMC de 1994.

[128] The Linux Information Project. (2005). *Algorithms: A Very Brief Introduction*. Obtenido de *LINFO:* http://www.linfo.org/algorithm.html

[129] En el caso *Whelan v. Jaslow* (revisado más adelante), el 3rd Circ. sostuvo que: "existen varias maneras en que la misma información puede ser organizada, dispuesta, y utilizada por un computador" (traducción propia).

La primera disposición reza: “los programas de ordenador están protegidos como obras literarias en el marco de lo dispuesto en el artículo 2 (1) del Convenio de Berna. Dicha protección se aplica a los programas de ordenador, cualquier que sea su modo o forma de expresión”. El art. 10 del ADPIC sostiene una idea similar: “los programas de ordenador, sean programas fuente o programas objeto, serán protegidos como obras literarias en virtud del Convenio de Berna (1971)”.

Nuevamente, se trata de una importante clarificación en relación al estado del sistema que consagra nuestra LPICh, en tanto no se expresa internamente de manera inequívoca que los programas son protegidos como obras literarias[130]. Este es un aspecto relevante por la naturaleza ambivalente de un programa como código y resultados de cómputo o procesos.

Ni la LPICh o la Convención de Berna definen con precisión “obra literaria”, limitándose a un catálogo ejemplar. Interesa destacar en dicho catálogo que una obra literaria puede definirse por contrapartida a una obra audiovisual, y no se reduce a las obras del *género literario* (novelas, poemas, etcétera). Según el 17 U.S. Code § 101, las obras literarias son “obras, distintas a las obras audiovisuales, expresadas en palabras, números, u otros símbolos o signos verbales o numéricos, sin importar la naturaleza del objeto material (…) en el cual están fijados”. Luego, la jurisprudencia norteamericana del caso *Exxon Corporation v Exxon Insurance Consultants International Ltd.* (1982, RPC, “Exxon”), suplementa la definición con “*para ofrecer información, placer o enseñanza al lector”*. Debe observarse que este nuevo elemento, relacionado con el requisito de fijeza de la obra a efectos de ser percibida genera problemas para articular la protección del código objeto que es, ininteligible para, probablemente, la mayoría de los humanos.

Que un programa se proteja como obra literaria implica que el concepto legal de programa alude a su faz escrita o de código, su primera fase de hecho, lo que es consistente con el art. 1 de la LPICh que dispone que los derechos nacen “por el solo hecho de la *creación* de la obra”. Las fases siguientes que atraviesa el código tienen fines principalmente utilitarios: modificar y preparar el código fuente para que produzca resultados.

[130]El artículo 75 inciso 1° de la LPICh dispone que “en el momento de inscribir una obra en el Registro (…) se depositará un ejemplar completo, manuscrito, impreso o reproducido. Tratándose de obras no literarias, regirán las siguientes normas (…).
El reconocimiento implícito está en que, al contemplar el catálogo de obras no adscritas al registro aplicable a obras literarias, el artículo no aborda o menciona a los programas.

Respecto de la protección de la faz de proceso de un programa y los resultados de cómputo, se debe descartar desde ya la protección de *las funcionalidades en sí*. Sin perjuicio de lo anterior, el estado del procesamiento de la información por los computadores modernos permite que la ejecución de un programa sea representada mediante resultados audiovisuales perceptible fácilmente y de alta sofisticación (ej. imágenes, sonidos, música). Estos aspectos pueden perfectamente entrañar obras susceptibles de protección, pero comprenderlas como *obras literarias* es forzoso. Según la misma OMPI, sucede que los resultados audiovisuales se hallan amparados por el derecho de autor pero de manera separada al programa, vale decir, no entran en el concepto legal de programa pero si en el de obra audiovisual general. Se adelanta que la protección de los resultados audiovisuales, por su relación al desarrollo de los programas se abordará solo someramente en este ensayo.

6.1.2. Ley 17.336.

El artículo 3 N°16 sostiene que "quedan especialmente protegidos con arreglo a la presente ley: 16) los programas computacionales, cualquiera sea el modo o forma de expresión, como programa fuente o programa objeto, e incluso la documentación preparatoria, su descripción técnica y manuales de uso".

Esta disposición armonizada con las disposiciones del Tratado OMPI y el ADPIC, da por sentado que los programas son protegidos en su faz escrita o código. Como dato adicional, según la OMPI (2007), las oraciones "sea en código fuente o código objeto" y "cualquiera sea el modo o forma de su expresión", son redundantes, observando que la segunda fórmula es más útil ante la eventualidad de que "en el futuro la categorización código fuente/código objeto devenga en obsoleta".

Cabe decir que la redundancia que aludía la OMPI es solo *en cierta medida* efectiva. Comprendiendo que el término "forma de expresión" alude fundamentalmente al lenguaje de programación, el cual puede ser así de cualquier clase (ej. cualquier lenguaje de alta abstracción existente, incluso lenguajes de baja abstracción como el lenguaje ensamblador, incluso, y a falta de exclusión, el lenguaje de la máquina de 1s y 0s). Sin embargo, es importante que la ley disponga inequívocamente que los programas se protegen como *código* debido a su naturaleza ambivalente (código o resultados). Es curioso que esta categorización "código fuente/código objeto" se utilice de manera impropia por la mayoría de los legisladores, incluyendo el nuestro. Debe clarificarse que el concepto técnico de *código objeto*, interpretable según lo dispuesto por el art. 21 del Código Civil, alude al código resultante del proceso de compilación y no necesariamente al código de la máquina. Igualmente técnico es el

concepto de *código fuente* y que alude al código como originariamente se ingresa, el cual puede, en casos extremos de programadores expertos, ser incluso directamente en lenguaje de la máquina. Por el hecho de que la LPICh protege uno u otro "cualquiera sea el modo de expresión", la interpretación técnica sería redundante y terminaría siendo igual de amplia a la manera probablemente dispuesta por el legislador.

La protección del código objeto es probablemente un aspecto controvertido y que, a mi juicio, evidencia el hecho de que al proteger los programas, los sistemas de derecho de autor sufrieron una especie de hibridación considerable. Según Hollaar (2002)[132], uno de los motivos sería que el código objeto, especialmente cuando se encuentra almacenado en memoria de solo lectura no modificable o ROM (ej. algunas clases de firmware), *puede* considerarse como una parte física (hardware) no susceptible de protección, en tanto termina identificándose con dicho aspecto físico, no siendo una "obra intelectual". Este argumento es flagrantemente débil en razón de que el hardware y el software, guardarán siempre alguna clase de interdependencia ya que el software no montado en hardware no existe en esta realidad. Que esta interdependencia se vuelva más intensa en ciertos casos (ej. el firmware) no debe generar dudas sobre el carácter necesariamente intangible de los programas.

Un argumento más válido para cuestionar la protección del código objeto, siguiendo la teoría implícita en Exxon, es que este no puede ser protegido como auténtica obra al no estar destinado a ser comunicado ni entendido por un humano y si por una máquina, de modo que dicho código sigue propósitos fundamentalmente utilitarios (operar una máquina) o porque derechamente no satisface un requisito de elegibilidad. Ni la Convención de Berna en su artículo 2(1), ni nuestra LPICh en el artículo 1, contemplan, en general, la exclusión de obras solo por el hecho de que persigan fines utilitarios, primando la protección de las obras eligibles cualquiera sea la forma o modo de expresión. En realidad, esta observación del carácter utilitario de los programas no es especial del código objeto, ya que el código fuente también persigue fines utilitarios (procesarse en código objeto). Tal como observa la OMPI, esta objeción descuida el hecho de que un programa puede traducirse "de código objeto a código fuente", en un proceso a veces llamado descompilación y que genera un grado de identidad entre ambos códigos. Se estima que, de querer articular una argumentación a partir del mismo derecho de autor para proteger el código objeto, el argumento de la OMPI (identidad de las obras) es quizás suficiente,

[132] Ob. cit. supra nota [126].

pero no es totalmente convincente. Otro argumento, al menos aplicable en nuestro sistema, es que simplemente no existe una regla análoga (explícita o implícita) a la regla de Exxon.

Producto de que una de las expresiones más comunes del intercambio de información en la industria tiene por objeto comercial al código objeto y no al código fuente (ej. software propietario), es que proteger de manera expresa al código objeto se vuelve una necesidad de proteger los intereses jurídicos legítimos de una gran cantidad de actores de la industria misma. En nuestro sistema, de hecho, la protección del código objeto fue anterior a la del código fuente, al menos en lo que respecta a la consagración expresa. En cualquier caso, como la legislación es expresa al incluir los conceptos de "código objeto" y "cualquiera sea el modo o forma de expresión", y sentada la importancia económica de la consagración, resultará estéril entrar en más detalles y discusiones: existe protección directa del código objeto, ininteligible que sea. Esto evidencia, a mi juicio, que los programas tal vez si merecen (en realidad *merecían* al momento de las discusiones relativas a la consagración de un sistema sui generis de protección) una cierta protección especializada y que se satisface con la inclusión del código objeto.

Sentada la necesidad de la protección del código objeto, es una situación totalmente distinta el hecho de que el código objeto constituya o no una *obra derivada* del código fuente. La compilación del código para su ejecución consiste en su literal traducción de un lenguaje de programación a otro, de modo que es posible comprenderlo como un acto de adaptación de al tenor de los arts. 5 letras i y w y 18 letra c. Sin perjuicio de esto, nuestra LPICh opta por generar una identidad entre el código fuente y el código objeto, de modo que la compilación o descompilación, si se hace entre estas dos códigos, es una suerte de copia o reproducción de la obra.

Si bien la frase "cualquiera sea el modo o forma de su expresión" aplicada a los programas parece referirse al lenguaje de programación utilizado, también existen maneras alternativas de interpretarla. El considerando (7) de la Directiva 2009/24/EC de la Unión Europea sobre Protección Legal de los Programas Computacionales ("Directiva UE 2009" o "Directiva UE sobre Programas") parece implicar que los programas son protegidos como código, independiente del medio en que se hallan reproducidos (ej. dispositivo digital v/s en un libro). Tal marco parece extender el concepto de "modo o forma de expresión" también al *medio* o soporte utilizado, el cual puede ser, según la disposición comentada, de cualquier clase, algo que está en cierta medida amparado por el derecho de reproducción consagrado en todo sistema y que en nuestra LPICh es suficientemente amplio (el art. 5 letra u se extiende por o a

“cualquier medio o procedimiento”). El problema de esta interpretación radica en que extiende el alcance técnico de lo que se entiende por programa como ámbito funcional de un computador físico, vale decir, el carácter esencialmente digital de un programa y que se aborda en el art. 5 letra t) que se verá.

En realidad, la extensión del alcance técnico de programa no es algo de por sí descartado en la LPICh. Relacionado con esto, nuestro legislador, como muchos otros (ej. la misma Directiva UE 2009) deja asentada la protección de documentación de programas, extendidas a la documentación preparatoria, manuales de uso, y descripción técnica, a propósito o en relación a la protección de los programas. Ubicar la protección de este tipo de ámbitos vinculándolos expresamente al concepto legal de programa es de simple técnica legislativa ya que, tal como advierte la OMPI (2007), dichos ámbitos pueden encontrarse protegidas según el concepto genérico de obra literaria (versiones incompletas del programa) o incluso artística (ej. imágenes promocionales).

¿Cuál es el alcance del concepto de programa en nuestra ley? El art. 5 letra t de la LPICh define programa como “un conjunto de instrucciones para ser usadas directa o indirectamente en un computador a fin de efectuar u obtener un determinado proceso o resultado, contenidas en un casete, diskette, cinta magnética u otro soporte material”, y a su vez, la copia de un programa consiste en un “soporte material que contiene instrucciones tomadas directa o indirectamente de un programa computacional y que incorpora la totalidad o parte sustancial de las instrucciones fijadas en él”.

Es posible puntualizar como antecedente de esta exacta definición, el informe de 1978 de la CONTU (pp. 30), marco que dispuso y preparó el concepto de programa que existe desde 1980 en el 17 U.S. Code. En uno u otro caso, la comprensión de los programas alude a su definición técnica: información funcional en soporte digital. No es posible comprender aquí la información digital (concepto amplio de software) que no sea de índole funcional (concepto de software restringido únicamente a los programas), algo que quizás sea obvio (ej. un libro digital se protege como libro y no como programa, por más que exista necesariamente un programa a efectos de, por ejemplo, descomprimir y leer el formato del archivo y luego visualizar el texto en cuestión). La primera advertencia es que el concepto de programa propuesto por el legislador implica extender la protección a toda clase de programas, sin entrar a discriminar según la funcionalidad o funcionalidades que ofrecen (ej. sistema operativo,

aplicaciones), ámbito que es, tal como comprendía el art. 2 del Tratado de la OMPI de 1996 vigente en Chile, extraño al derecho de autor.

En el curioso caso *Apple Computer, Inc. v. Franklin Computer Corp* (3d Cir., 1983, "Franklin"), se intentó, sin éxito, argumentar que un sistema operativo no es siquiera *elegible* por la vía de derecho de autor por una seguidilla de argumentos, todos descartados: a) por entenderse estos como "métodos" o "procesos"; b) por tener una naturaleza utilitaria; y c) por existir aplicación de la doctrina de la fusión idea expresión; de manera que, en general, se intenta sostener que un sistema operativo es un fenómeno solo susceptible de un derecho de patente.

Para el tribunal tales argumentos perdían completamente el foco de la pretensión deducida, ya que el demandante no alegaba la protección de los métodos o procesos yacentes en el programa (las funciones) de modo que esta discusión introducida por el demandado era impertinente. Los problemas inherentes en la protección de los programas, especialmente su naturaleza utilitaria, se extienden a toda clase de programas y no de manera especial al *software sistema*. La doctrina de la fusión fue desestimada por que fue impertinentemente esgrimida por el demandado pero puede utilizarse la argumentación y comentarios propuestos al tratar el Tratado de la WIPO de 1996 en su artículo segundo. Quizás el único argumento que era necesario para desestimar los argumentos vistos en *Franklin* radica en que la ley, al ofrecer un concepto amplio de programa, no entra a discriminar la extensión de la protección mediante derecho de autor según la o las funcionalidades que ofrece. Nuevamente, la funcionalidad de un programa es un ámbito extraño al derecho de autor, no así, el potencial creativo presente en su desarrollo.

Una segunda advertencia es que no importa la extensión o complejidad del programa. El concepto de programa no discrimina según funcionalidad ofrecida o según la extensión del código (ej. cantidad de líneas), aunque si alude a que exista "un conjunto", aunque es cierto que programas muy escuetos tendrán menos chance de ser protegidos al conllevar, usualmente, menos esfuerzo creativo. Relacionado al diseño modular en los programas, un programa particular sacado al mercado se compondrá de un conjunto de programas más pequeños y concretos, todos interconectados. Por ejemplo, los procesadores de texto actuales ofrecen funciones de escritura y edición de textos, cambio de formato de archivos, guardado e impresión de archivos, y un enorme etcétera. Bajo estas funcionalidades un poco más concretas, posiblemente subyacen otras funciones aún más concretas y

detalladas, usualmente cada vez más técnicas, y así sucesivamente[134]. Sea una funcionalidad más general u otras más concretas o delimitadas, constando que cada una está dada por un programa, es que cada uno de estos "miniprogramas" es una obra potencialmente elegible de protección y, a mi juicio, lo es de manera autónoma[135]. Llevada al extremo esta idea, y no viendo real inconveniente, diríamos que un gran grueso de programas, que el usuario común entiende como tales[136], y a la luz del derecho de autor, son auténticas compilaciones de obras protegidas.

La última advertencia radica en que los adjetivos "directa o indirectamente" se dispusieron para hacer indudable que el código fuente se encuentra protegido, habida cuenta que en nuestra LPICh se define programa en su visión técnica. Solo el código objeto ejecutable tiene la aptitud para su utilización directa con el computador, vale decir, instruirlo. El código fuente generalmente debe traducirse, de ahí la palabra "indirectamente". Ámbitos como el simple diseño preparatorio del programa (ej. escritura o planificación temática de las funciones o algoritmos presentes en programa *en lengua natural*) no entran en el concepto de programa al no tener dicha aptitud ni aún tras un procesamiento adicional, aunque se pueden encontrar protegidos de acuerdo a las reglas generales.

Si el concepto legal de "programa" en la LPICh es clarificador y acorde con la técnica, el concepto de "copia de programa", que alude al soporte material o ámbito tangible, es contraproducente[137]. A principios de la década de 1990 cuando entró en vigencia la Ley 18.957 que estrenó la disposición en comento, el *software de retail* basado en distribución de software contenido en copias tangibles era ciertamente un fenómeno relevante. En ese sentido, el concepto y la idea de "copia" es mencionado de manera no sistemática y no coherente a propósito de diversos derechos de autor, como el de distribución y reproducción, pero terminar por obstaculizar ciertas interpretaciones del alcance de cada

[134] En el ejemplo del procesador de texto, en la funcionalidad de cambio de formato de archivos posiblemente yacen funciones de compresión de archivos. Luego, sobre la función de compresión yacen probablemente funciones matemáticas y algoritmos de ordenamiento, solo por mencionar algunas.

[135] En el caso *Sega v. Accolade* (1992; 9th Circ.), el tribunal sostuvo que "en realidad, la función principal o propósito de un programa es el resultado compuesto de subrutinas interactuando entre sí (…) en base a un método que descompone un programa en las rutinas y subrutinas que lo componen y que luego identifica la idea (…) *de cada uno* (…).

[136] Los usuarios generalmente reservan el concepto de programa para verdaderos productos complejos, vale decir, un conjunto de funciones esperables a efectos de resolver problemas prácticos (ej. un *office suite*, como el programa utilizado para escribir este ensayo). Con todo, cuando los desarrolladores al concepto de programa, generalmente aluden a funciones más específicas y técnicas (ej. si el desarrollador en cuestión es precisamente el desarrollador de un office suite, quizás esté interesado en un programa que contenga algoritmos para dar formato a archivos computacionales, un programa generador de llaves de activación, etcétera).

[137] Es interesante el hecho de que en el recorrido del art. 5 de la LPICh no existe definición general de "copia de obras protegidas", sino exclusivamente como concepto aplicable con respecto a los programas.

uno[138]. Cualquier usuario puede obtener copias de programas en su *disco duro* mediante operaciones conocidas ("copiar-pegar"), pero estas copias no se pasan a identificar con el dispositivo físico, salvo que se reduzca al absurdo el concepto legal de copia (ej. la "copia del programa" *es* el disco duro). Tal como el programa matriz utilizado para obtener copias, que puede ser también una copia, estas copias son información digital y no otra cosa. Es incontestable que el contenedor o formato en que se almacene un programa no cambia su esencia: es siempre inmaterial (información). A esta altura de la historia (quizás no tanto al año 1990), la independencia comercial de la industria de software con la de hardware es demasiado notable como para confundirlas.

6.2. Expresión original, resultados audiovisuales y elementos no literales.

Las obras protegibles *relacionadas* a los programas son: el código (como obra literaria), los resultados de cómputo o displays (como obra audiovisual) y otras obras (literarias o audiovisuales) que existen, comúnmente, aunque no siempre, a propósito del código o los displays (ej. documentación preparatoria, imágenes promocionales). Sin perjuicio de esto, cuando se profiera el concepto de programa, al menos en el contexto de la LPICh, se está apuntando al código y no a otra obra. Esta es la obra cuyos contornos jurídicos se busca delimitar con precisión.

Para determinar el alcance particular del derecho de autor en los programas es imprescindible desentrañar que se entiende por expresión original. En realidad, cuando se habla de la protección jurídica de los programas por el derecho de autor hablamos más concretamente de proteger la expresión original (si la hay) subyacente. En cierta medida, este concepto se relaciona a *la forma particular en que el programador implementa* las distintas funciones del programa (y no las funciones en sí), la cual entonces debe ser original. El adjetivo original apunta a que dicha forma no debe ser una copia de otra expresión ya protegida.

Esto implica que en los programas se requiere de un análisis especialmente fino para determinar qué entra propiamente al sistema derecho de autor para ser protegido de lo que no lo es. En general, habida cuenta que los intensos e ineludibles ámbitos funcionales del programa, considerados autónomamente, no son objeto de derechos de autor, es que se dice que la aptitud protectora es menor comparada con

[138] La más típica es la subsunción de la distribución digital (de toda clase de obras, incluyendo a los programas) dentro del concepto legal de distribución que alude el art. 5 letra q de la LPICh.

la existente en otra clase de obras más tradicionales, situación que la jurisprudencia norteamericana conoce como *thin (scope of) protection*. En realidad esto puede comprenderse como una especie de necesaria coordinación de los medios de protección potencialmente implicados en un programa, ya que un programa también puede ser susceptible de protección por medio de patentes y secretos empresariales. Por esta posibilidad de concurrencia protectora que ocurre en los programas, los distintos medios jurídicos deben procurar no extenderse sobre los ámbitos que le competen a otro. Otra manera de comprender este *thin scope of protection* apunta a que la libertad creativa de un programador es mucho menor a la que tiene, por ejemplo, un poeta o un escritor de ciencia ficción.

6.2.1. Resultados audiovisuales de un programa. Breve mención.

Según los términos del art. 3 N° 16 de la LPICh y los arts. 2 y 4 del Tratado OMPI de 1996, las *displays* de un programa pueden protegerse por el derecho de autor como obra audiovisual y no como el programa en su concepto legal de obra literaria, vale decir, el código.

La sola admisión de que las *displays* de un programa se hallan protegidas, es posible datarla de manera ya cohesionada a partir de la década de 1980, al menos a propósito de un videojuego y en la jurisprudencia norteamericana, no sin algunas dificultades, presentada en los casos *Stern Electronics Inc. v. Kaufman* (1982), *Midway Manufacturing Co. V. Artic International, Inc.* (7th Cir., 1983, "Midway"), entre otros. También se reconoce la protección de las interfaces gráficas de usuario (ej. ventanas, bordes, íconos, colores) en el caso *Broderbund Software Inc. v. Unison World, Inc.* (N.D. Cal., 1986, "Unison")[139]. En dicho sistema, se define una obra audiovisual como "obras que consisten en una *serie de imágenes* relacionadas que están intrínsecamente destinadas a ser percibidas con el uso de máquinas (…) juntamente con los sonidos que las acompañan, si hubieren, sin importar la naturaleza de los medios materiales (…) en que dichas obras están fijadas". Advirtiendo la potencial falta de aplicación de la disposición citada al caso de las displays, producto de que el flujo de imágenes y sonidos lo controla el usuario, es que el tribunal del caso *Midway* decide hacer una interpretación extensiva, comprendiendo una obra audiovisual con concepto propio: "cualquier conjunto de imágenes mostradas como una unidad". Convendría también agregar aquí la frase "con o sin sonido". A efectos de nuestro sistema, ni la LPICh o el Convenio de Berna define obra audiovisual, aunque la Ley 20.243 menciona

[139] Sobre esto, relaciónese más con los "elementos no literales individualmente considerados" a que alude el caso *Blizzard* visto más abajo.

importantes normas aplicables a esta clase particular de obras, por lo que no existe obstáculo legal alguno para negar potencial protección de displays, habida cuenta que la enumeración del art. 3 de la LPICh es meramente enunciativa.

También expresa en *Midway*, está la conclusión quizás necesaria de que la actividad del usuario en la obtención de las *displays* (la literal ejecución del programa) no es una actividad creativa del primero[141], ya que "jugar un videojuego se asemeja más a cambiar los canales en un televisor que a escribir una novela o pintar una pintura (…) el videojuego efectivamente escribe las oraciones y pinta la pintura por él; el usuario simplemente elige una de las oraciones ya fijadas en la memoria, una de las pinturas almacenadas en el catálogo". Advirtiendo este análisis, en el año 2015 se intentó implicar que la actividad del usuario podría comprenderse como una *obra derivada* de la obra original (display de programa), y que tuvo como protagonistas a dos plataformas de *streaming* (Twitch y Azubu) y a un jugador profesional, pero ningún precedente judicial.

Lo relevante respecto de la protección de los resultados audiovisuales, y del porque se reserva un acápite en este ensayo aparentemente abocado al análisis del código, consiste en que dicha protección se encuentra, si bien relacionada, efectivamente separada de la protección que afecta al código del programa. Esto, a mi juicio, resulta conveniente a efectos de no confundir todos los aspectos en juego susceptibles de protección *a propósito* de un programa, que terminan alcanzando, al menos, al código y los resultados audiovisuales. En ese sentido, no se comparte el comentario inserto en la Circular 61 de la *Copyright Office*[142] de los Estados Unidos que trata sobre el registro de los programas y otras obras relacionadas, en cuanto dice que "un programa y los resultados de la pantalla son generalmente considerados como *una misma obra* porque la mayor parte de dichos resultados son creados por el código del programa", pudiendo deberse este comentario a la idea de que los *displays* constituyen una reproducción del código, algo que tiene cierta verdad técnica. Sin perjuicio de esto, en este ensayo se aboga por comprender que las *displays* se relacionan al programa, pero su protección está separada de este en tanto son *obras* de distinta naturaleza y, por regla general, creadas por distintas personas. Sobre las displays, por ejemplo, no hay que entrar a discutir la "naturaleza funcional de la obra" presente en el código. Además, las displays comúnmente consisten en la interacción, más que del código, de obras

[141]Véase por ejemplo, el caso *Fortnightly Corp. v. United Artists* (SCOTUS, 1968).
[142]Documento público, disponible en línea: https://www.copyright.gov/circs/circ61.pdf, consultada el 15 de marzo de 2020.

tradicionales protegidas de naturaleza distinta a este (ej. imágenes, composiciones musicales). Estas obras aludidas no son creaciones de un programador sino que de artistas gráficos, artistas de 3D, músicos, etcétera.

Ya desde el caso *Manufacturers Technologies, Inc. v. CAMS Inc.* (1989) se consideraba, no obstante su obvia interrelación, que en un programa podían existir al menos dos aspectos protegibles claramente distintivos. Esta separación de los aspectos susceptibles de protección a propósito de un programa se resume más sistemáticamente en el caso *MDY Industries, LLC v. Blizzard Entertainment, Inc. and Vivendi Games, Inc.* (2010), distinguiéndose por el tribunal[143] de hecho tres, y no dos, elementos distintivos, cada uno susceptible de protección autónoma, a propósito de un "producto de *software"*: a) sus elementos literarios, dados por el código; b) los elementos no literarios individualmente considerados, dados por los distintos componentes visuales o auditivos utilizados (ej. composiciones musicales, imágenes); y c) los elementos no literarios considerados en su todo (o en palabras del tribunal, "dinámicamente"), dados por "la experiencia en tiempo real" que experimenta el usuario. Estos elementos no literales en su forma dinámica son sino los resultados audiovisuales de un programa (que en el caso *Blizzard* puede definirise más someramente como que el *videojuego en sí* es una obra protegida) y resultan sino una obra separada (ej. se debe registrar, si así se desea, aparte del código), tanto del código como de las otras formas de expresión utilizadas para conseguir dichos resultados y que aquí se denominan como elementos no literarios.

6.2.2. Código del programa: elementos literales y no literales.

La protección de la LPICh en los programas en cuanto tales se restringen al código como obra literaria y su alcance es a priori menor si se compara al ámbito de protección otorgado a obras más tradicionales como un guion cinematográfico.

Sobre la determinación más precisa de en qué consiste *la expresión original* en el código, una de las tendencias asentadas en los distintos casos de la jurisprudencia norteamericana, consiste en ciertamente en proteger la *literalidad* o el código letra por letra (que permite accionar fácilmente contra quienes copien textualmente el código), pero a la vez, y tras el uso de ciertos métodos, extender dicha protección a aspectos "no literales" como puede ser la estructura del código en sí. En uno de los casos

[143]Tribunal de los Estados Unidos para el Distrito de Arizona. Luego el *9th Circ.* pasa a utilizar idéntica sistematización al conocer de las apelaciones.

que se analizan más abajo, el tribunal fija la base argumentativa de este mayor alcance protectivo sosteniendo que "uno puede vulnerar el derecho de autor en una dramatización o libro copiando la trama o sus recursos de trama". En el otro caso analizado, el tribunal sostuvo que "en analogía a *otras obras literarias*, pareciera ser que los derechos de autor sobre programas pueden ser vulnerados incluso no habiendo copia de los aspectos literales de los programas".

Aplicada en realidad a toda clase de obras y no solo a los programas, la determinación de los elementos no literales es una tarea eminentemente jurisprudencial, de ponderación y balance de intereses, donde los hechos del caso toman preponderancia así como los principios y filosofías implícitas de cada sistema (ej. teoría utilitaria). Los métodos para la determinación de estos esquivos elementos no literales serán no lineales, siempre perfectibles, y es poco probable que exista o, se llegue en algún momento, a una única solución metodológica (Samuelson, 2013)[144]. Su fundamento radica en garantizar un justo medio de protección que genere suficientes incentivos para el desarrollo de obras creativas sin reconducir a conferir monopolios sobre ideas (y en el caso de los programas, de funciones), por lo que parece ser que la protección de elementos no literales es una cuestión de equidad.

La protección de aspectos como la estructura del código resultan razonables, ya que la programación no pone tanto énfasis en el código en si, sino en principios de desarrollo (ej. control de la complejidad, diseño modular, algoritmos eficientes, elegantes y robustos, etc.), y en último término, en satisfacer necesidades humanas. En ese sentido, si nuestra jurisprudencia adopta la filosofía utilitaria de la propiedad intelectual (Menell, Lemley, & Merges, 2019)[145] como fundamento del derecho de autor nacional, hará bien en extender la protección a elementos más allá de la simple literalidad del código. Siguiendo a Hollaar (2002)[146], es posible distinguir distintos "métodos" jurisprudenciales que buscan determinar el alcance de los elementos no literales. Más que "métodos para determinar los elementos no literales" se trata de hitos y ámbitos clarificadores necesarios con respecto a la forma concreta en que el derecho de autor protege a los programas para determinar aspectos tan importantes como si hubo o no infracción en casos concretos. Mejor aún, estos métodos permiten comprender que se

[144] Samuelson, P. (2013). *A Fresh Look at Tests for Nonliteral Copyright Infringement. Northwestern University Law Review (Vol. 107, No. 4)*, pp. 1822-1823; 1837-1839; 1842-1843; 1847-1849.
[145] Menell, P. S., Lemley, M. A., & Merges, R. P. (2019). *Intellectual Property in the New Technological Age: 2019.* Clause 8 Publishing, pp. 16-22.
[146] Ob. cit. supra nota [126].

protege y que no en un programa, vale decir, hallar el alcance preciso de la *expresión original* en un programa. En este ensayo se esbozan y analizan los dos métodos principales.

A) ESTRUCTURA, SECUENCIA Y ORGANIZACIÓN ("STRUCTURE, SEQUENCE AND ORGANIZATION" O "SSO").

Aplicada primitivamente en el caso *Whelan Associates v. Jaslow Dental Laboratory* (1986, 3th Circ.) y fuertemente basada en lo sostenido por la Suprema Corte de Justicia de los Estados Unidos en el caso *Baker v. Selden* (SCOTUS, 1879), bajo el SSO se entiende que todo elemento que no se identifica *con* las ideas de la obra o que no son esenciales para desarrollarlas, que en las obras de tipo funcional o utilitaria como los programas reconduce a sus funciones, forma entonces parte de la expresión protegible. En esta jurisprudencia los conceptos de "estructura", "secuencia" y "organización" son en realidad intercambiables ya la idea principal dice tan solo que "la secuencia y orden *pueden ser parte de la expresión*, no de la idea, de la obra".

Si la idea es la función del programa, la expresión consiste en "la forma en que el programa opera, controla y regula al computador". Dentro de la expresión se incluirá ciertamente el texto del código y a su vez, entendiendo que "uno de los costos más significativos presentes en la creación de programas radica en el desarrollo de la estructura y lógica", es que *pueden* incluirse: estructuras de datos, resultados audiovisuales del programa, y funciones[147].

El concepto técnico de "función", también a veces mencionados como "métodos", se relaciona con el diseño modular tantas veces mencionado: bloques particulares del código general, empaquetados como una unidad lógica (ej. con un nombre particular) generalmente en atención a la funcionalidad provista (ej. "función ObtenerDistancias"). Dentro de un programa existirán componentes funcionales (que instruyen al computador y que por ende se pueden considerar como un programa dentro de otro programa), verdaderos ladrillos de una construcción más grande que es la aplicación final, y otros más de índole estructural, estos últimos sirviendo como ámbitos de interoperabilidad entre componentes que si son funcionales, vale decir, el cemento o argamasa que une a los ladrillos.

La regla SSO en la forma del caso *Whelan* no tardó en ser criticada por su impertinente extensión, análoga a la protección de funciones de un programa por patentes (Hollaar, 2002)[148]. Incluir protección

[147]En *Whelan*, el tribunal utilizó para arribar a su decisión precisamente estructuras de datos, resultados audiovisuales y unas cuantas funciones bien precisas del programa general.
[148] Ob. cit. supra nota [126].

de manera generalizada a “estructuras de datos” o “funciones”, sin entrar a analizar en qué consisten concretamente dichos aspectos lógicos, vale decir, hacer un análisis propio sobre si en estos aspectos más concretos existe expresión original propio[149], puede perfectamente extender el ámbito del derecho de autor a ideas. Otro aspecto problemático consiste en incluir o pretender incluir la protección de los resultados audiovisuales como si fueran una “obra literaria no literal”[150]. Una explicación del increíble alcance de *Whelan* es que, por los hechos del caso, no se intentó ahondar mucho en la argumentación jurídica, habida cuenta que era hecho probado que el demandado había tenido contacto directo con el código fuente del demandante.

En ese sentido, en el caso *Computer Associates v. Altai* (próximo caso a analizar), se crítica la jurisprudencia presente en Whelan por estimarse que no atiende a las consideraciones más prácticas de la estructura de los programas, vale decir, a la forma efectiva en que se desarrollan los programas (o la mayoría de estos): el diseño modular. En el caso *Gates Rubber Co. v. Bando Chemical Industries, Ltds.* (1993, 10th Circ.) el tribunal también comprende que la estructura de un programa es “una descripción de cómo funciona el programa según sus múltiples funciones, las cuales son realizadas por *módulos discretos*, así como la forma en que cada uno de estos módulos interactúa con el otro”, saltando a la vista como existe acogida ya casi expresa del diseño modular. *Whelan* es criticable porque, por el hecho de que cada módulo es en sí un programa con su propia función particular, por ende con su propia “idea”, es que en un programa pueden existir en realidad varias ideas no siendo posible decir que la expresión en un programa es “todo aquello” que no se identifica con la idea del programa final. Con todas las posibles críticas, el caso *Whelan* se alza como una jurisprudencia pionera en el ámbito de proteger ámbitos *extra literales* en los programas e incorpora la posibilidad de que la estructura del programa final es un ámbito susceptible de protección.

Relacionado a esto, en el aún no resuelto caso *Oracle America, Inc. v. Google, Inc.*, se menciona como el *diseño modular* (que es simplemente una práctica usual en la programación) se relaciona a la estructura y por ende, a un posible ámbito de protección de los programas. En este caso, Oracle demanda a Google por presunta infracción de derechos de autor respecto de dos conceptos: por copiar

[149]También resultó muy notable para el tribunal, *la similitud de la estructura* de los programas implicados. Sin perjuicio de esto, uno de los peritos presentados por el demandado sostuvo que “en realidad, hay pocas estructuras de datos eficientes posibles, por lo que la similitud de las estructuras presentes en los programas no resulta ni sorpresiva o particularmente incriminadora” (traducción propia).

[150]Adviértase que los elementos no literales lo son con respecto a la *obra literaria*.

ámbitos estructurales de código del cual tiene derechos de autor[151] y por copiar de manera literal nueve líneas de código, estas últimas las cuales comprenden *un módulo o función* en la forma definida en este ensayo. Una posible comprensión del caso *Oracle* apunta a estimar que el demandante busca proteger la estructura del código por la vía de obra colectiva o compilación (ej. art. 3 N° 17 de la LPICh protege la disposición -estructura- de los contenidos en una compilación de obras protegidas) y un programa particular dentro o presente en dicha compilación. Esta comprensión es posible de sostener, en parte, porque ni el 17 U.S. Code, ni nuestra LPICh en caso de que se quisiera aplicar esta teoría en nuestro país, niega la protección de un programa por el hecho de ser escueto o no tan complejo.

Del caso *Whelan* también es posible rescatar sus aspectos prácticos, principalmente los probatorios. Por la complejidad y extensión de los programas se hace perentorio incluir de manera precisa las partes del código que se estiman implicadas en una demanda de infracción de derechos de autor. Sobre cada parte habría que reservar pruebas de titularidad (o permisos, si corresponde), en caso de que esta sea cuestión controvertida en juicio. Estas partes o bloques particulares del código, deben comprenderse como un programa en sí susceptible de protección por la vía de derecho de autor porque nuestra LPICh no distingue. Eso sí, por más complejo o simple que sea un programa, debe existir expresión original, siendo quizás más fácil encontrarla en programas complejos y de muchas partes o módulos. Incluir los *displays* del programa en juicio, aún si no son objeto de este, producto de que son la manifestación más perceptible de los programas, puede ser útil como prueba circunstancial de una presunta copia del *código*. Otra cosa es que se busque fundar la demanda por supuesta infracción de derechos de autor respecto de los *displays* y no del código. Sin perjuicio de esto último, tal como sostuvo el mismo tribunal en el caso *Whelan*, aun teniendo en cuenta que los *displays* "deben tener alguna relación con el programa (como código)", pueden terminar por confundir al sentenciador y hacerlo presa de un prejuicio injusto (Moreno, 1990)[152][153]

[151]Oracle sostuvo que "*it is undisputed that Google copied 7,000 lines of declaring code and generally replicated the overall structure, sequence, and organization of Oracle's 37 Java API packages.".* El desarrollo de *application program interfaces* (APIs) es una expresión de programación común y consistente con el diseño modular y tal como su traducción dice, refiere a código fuente cuya función es servir de interface o conexión entre otros módulos si funcionales.

[152] Moreno, R. D. (1990). *"Look and Feel" as A Copyrightable Element: The Legacy of Whelan v. Jaslow? Or, Can Equity in Computer Program Infringement Cases Be Found Instead By the Proper Allocation of Burden of Persuasion. Louisiana Law Review (Vol. 51, No. 1)*, pp. 177-211.

[153] Pese a que el mismo tribunal advirtió la fuerza probatoria desproporcionada que tienen los resultados audiovisuales del programa (la jurisprudencia de *Whelan* parece seguir la visión de la Circular 61 de la *Copyright Office*), parece que igualmente

B) ABSTRACCIÓN, FILTRO, COMPARACIÓN ("ABSTRACTION, FILTRATION, COMPARISON" O "AFC").

El método AFC tiene de origen incluso terminológico en el caso *Computer Associates International, Inc. v. Altai*, Inc. (1992, 2nd Circ.), y es considerada jurisprudencia relevante a efectos de determinar si hubo o no copia de elementos no literales respecto de un programa (Samuelson, 2013), pero que, a diferencia de *Whelan*, pone mucho énfasis en las circunstancias prácticas de la programación y presenta una elaboración notable. Siguiendo a Hollaar (2002)[155], concurre a la determinación de los detalles del método AFC, la jurisprudencia del caso *Gates* mencionado más arriba.

B.1.) ABSTRACCIÓN.

En esencia, esta fase pretende hacer una lectura informada del programa tal como cualquier "obra literaria" (ej. una obra del género narrativo tiene un inicio, un desarrollo, y un final, luego cada parte tiene ciertos personajes, eventos, trama, etcétera). Esta lectura se hará determinando la estructura interna del programa, asilándose cada componente detectado, ya que el tribunal debe tomar en cuenta la manera *efectiva* en que los programas son típicamente creados (Ravicher, 2002)[156]. Debe tenerse en cuenta que, en general y parafraseando el título de una de las obras más conocidas al informático suizo Niklaus Writh (*Algorithms + Data Structures = Programs* del año 1976), los distintos elementos de un programa caen en dos categorías: elementos funcionales o algorítmicos y elementos más relacionados a la estructura o disposición de dichos elementos funcionales.

Sobre la fase de abstracción, el *2nd Circ.* postula el concepto de "niveles de abstracción". Este concepto refiere al *nivel de detalle técnico,* con respecto al programa final, presente en un elemento particular del programa. Mientras más se relacione un módulo o pieza estructural del código a la función última del programa (ej. una función, así como el código de interface, que detecte, controle y disponga los detonadores correspondientes ante las pulsaciones del teclado en un programa procesador de textos[157]) se dirá que tiene un *mayor* nivel de abstracción. El *2nd Circ.* notó así que la determinación de la

terminó cediendo, ya que les otorgó "amplio peso probatorio circunstancial" en la sentencia y sus consideraciones (Moreno, 1990: 205).

[155] Ob. cit. supra nota [126].

[156] Ravicher, D. (2002). *Software Derivative Work: A Jurisdiction Dependent Determination*. Obtenido de *Linux.com:* https://www.linux.com/news/software-derivative-work-jurisdiction-dependent-determination/

[157] En este mismo ejemplo, en dicha función pueden estar presentes algoritmos o funciones lógicas. Estas últimas, por relacionarse mucho menos al propósito principal del programa, se consideran entonces de menor abstracción.

estructura de un programa puede ser bastante compleja a los niveles más básicos de abstracción, pero deviniendo en "trivial" en los niveles más altos.

Comprender los distintos niveles de abstracción es equivalente a pasar por el código del programa con una especie de lupa o zoom reajustable, primero de gran amplificación (ej. análisis en detalle de una línea de código; análisis de una función compuesta por 15 líneas de código; análisis de tres funciones diseñadas para operar conjuntamente), y que termina con una vista panorámica de todo el programa. Como el método AFC se sujeta a la práctica (ej. diseño modular) el tribunal partirá por el código (nivel más bajo de abstracción posible, máximo detalle); pasará por las instrucciones precisas y los miembros o estructuras de datos de que se valen para operar; luego por las funciones o bloques de código compuestos, comprenderá la estructura o arquitectura del programa dada por las relaciones entre los demás componentes funcionales (ej. subsunción o *nesting*, dependencias); y desembocará finalmente en el propósito o función general del programa, pero de manera informada con respecto a cómo funciona, diríamos que el tribunal habrá "leído la obra".

La susceptibilidad de hallar "expresión" se reduce conforme subamos en los niveles de abstracción, siendo de hecho nula al tratar la funcionalidad principal del programa (que se filtra a priori) y muy alta en el nivel más bajo y literal: el código. A nivel del código mismo, tal vez lo único que hay que filtrar son los elementos y restricciones inherentes del lenguaje de programación utilizado. Hollaar (2002)[158] nota así que el método AFC cobra relevancia en los niveles intermedios de abstracción: instrucciones, estructuras de datos, módulos y la estructura general del programa. Por tener un sustrato ineludiblemente complejo, la abstracción, especialmente en niveles más bajos, requiere que el tribunal cuente con un bagaje pericial importante. Podemos decir que el método AFC comprende que el alcance de la expresión original los programas como obras de derecho de autor se debe informar, en cierta medida, con la disciplina de la programación.

B.2.) FILTRACIÓN.

Por esta fase se busca filtrar, de *todos los elementos que se abstrajeron*, aquellos aspectos que no son susceptibles de protección por la vía de derecho de autor. El tribunal hará la abstracción de cada elemento de manera separada, efectuando la filtración uno por uno, ya que se comprende que cada

[158] Ob. cit. supra nota [126].

elemento del programa es en cierta medida autónomo y candidato a protección (vale decir, poseer expresión original). Esta importante advertencia está implícita en el caso *Gates*[159].

La razones principales para filtrar refieren, en general, a que no puede decirse de manera razonable que el elemento fue implementado desde un abanico suficientemente amplio de posibilidades, vale decir, no es posible hablar de libertad creativa en la implementación del elemento. Desde ya se filtra el propósito principal o funcionalidad última del programa.

Los criterios para filtrar son:

1. El elemento ha sido implementado por razones de eficiencia:

El *2nd Circ.* sostuvo que (traducción propia):

"Uno puede considerar el hecho de que los programadores usualmente buscan crear programas que satisfagan las necesidades de los usuarios en la manera más eficiente posible (…) En el contexto de la programación, el concepto de eficiencia se relaciona con (…) formular la más computación matemática más concisa".

La pregunta relevante que se debe responder para realizar o no el filtro de eficiencia en el caso concreto, consiste en determinar si la implementación del elemento abstraído (usualmente de carácter funcional) obedeció o fue necesario por razones de eficiencia bajo el concepto propuesto arriba. El tribunal comprende agudamente que las consideraciones de eficiencia restringen la libertad creativa y terminan generando una identidad entre el elemento y su función, aplicándose la doctrina de la fusión. En este caso, el tribunal sostuvo que: "entonces la expresión representada por la elección de un módulo o grupo de módulo se ha fusionado con la idea subyacente y por ende no es susceptible de protección".

Nótese que el tribunal habla específicamente de los elementos denominados "módulos" o "conjuntos de módulos". El tribunal probablemente se refiere a los ámbitos funcionales del programa, en cualquiera

[159] Inspirado en doctrina estadounidense, el 10th Circ. sostuvo que (traducción y cursiva propias): "Un programa puede usualmente ser desmembrado en al menos seis niveles, generalmente decrecientes, de abstracción: (i) propósito general, (ii) estructura, (iii) módulos, (iv) algoritmos y estructuras de datos (v) código fuente, y (vi) código objeto (…) Luego de que un tribunal ha tenido éxito en identificar los distintos niveles de abstracción en un programa, *debe filtrar dichos elementos del programa* que no están protegidos por el sistema de derecho de autor".

Se sigue así que cada elemento se abstrajo para un análisis separado con respecto a los demás.

de sus capas de abstracción: desde una instrucción particular[160] a paquetes o bloques, más o menos complejos, de instrucciones, pudiendo pasar por la estructura o *esquema general* del programa implicado.

2. El elemento se ha implementado por factores o circunstancias externas:

Se entiende que dichos factores, circunstancias, consideraciones, externas limitan o restringen la "libertad" en la elección del diseño del programa y por ende no forman parte de la expresión.

En *Altai* se ofrecen algunos ejemplos concretos de estos factores externos (mera enunciación):

a) Requisitos de compatibilidad. Estos aluden sino al importante principio de interoperabilidad, sin distinguir si se refieren a especificaciones del computador físico sobre el cual se busca operar el programa (interoperabilidad hardware-software) o requerimientos de otros programas con los cuales el programa implicado debe operar conjuntamente (interoperabilidad software-software). El caso típico es la compatibilidad de un programa aplicación con los requisitos e interfaces propias de un sistema operativo determinado.

b) Estándares dispuestos por los fabricantes de hardware (puede reconducirse al punto anterior).

c) Necesidades propias o externalidades de la industria o mercado a la cual se está ofreciendo el producto o servicio de software.

d) Prácticas de programación ampliamente aceptadas.

Por la misma razón presente en estos ejemplos, es que deben agregarse las limitaciones del lenguaje de programación utilizado.

3. El elemento reside en el dominio público o se encuentra desprovisto de protección por alguna otra razón.

Este filtro, según dispuso el tribunal en el caso *Gates*, incluye también los procesos, funcionalidades, métodos de operación y otros sucedáneos, en tanto son ámbitos no susceptibles de protección por la vía de derecho de autor. Esto implica que el proceso en sí mismo y autónomamente considerado, se

[160]En el caso *Gates*, el tribunal utiliza el término "algoritmo" para referirse a una instrucción singular, lo cual conduce, generalmente (y por las limitaciones de los lenguajes de programación), a una línea precisa del código (sin perjuicio de las declaraciones).

halla desprotegido por ser más bien un ámbito extraño al derecho de autor, no obstante, la descripción que se haga del proceso o método por parte del programador, puede perfectamente hallarse protegida, en la medida que no sea una repetición mecánica del mismo (Hollaar, 2002)[161].

Entran para ser filtrados por este concepto, en cuanto a su contenido, los distintos algoritmos (matemáticos o no) de que se vale el programa, así como los lenguajes de programación. Un algoritmo no es sino un *proceso* que permite arribar a un determinado resultado. Un lenguaje de programación es, en esencia, un *sistema* de comunicación. Sobre estos dos aspectos, podemos citar, solo a modo ejemplar, el artículo 10 (3) Sección 1 del Capítulo 2 de la *Copyright Law of Japan*.

El mismo tribunal del caso *Gates* agregó también los hechos, datos y descubrimientos ("facts"). Con todo, la estructura, descripción o compilación que se haga de dichos hechos o datos puede hallarse protegida si constituye una expresión original (ej. estructura de bases de datos). Nuevamente, esto implica que el hecho en sí o autónomamente considerado no es susceptible de protección. Si el programador utiliza dicho hecho de manera creativa, no hay razón para no otorgarle protección.

Tal como advierte Hollaar (2002)[162] a propósito de este filtro, uno de los problemas de la fase de filtración consiste en que puede reducirse el sustrato de la fase siguiente a la filtración (vale decir, un "filtro por error"), prescindiendo de elementos y componentes del programa que, por constituir potenciales ámbitos de expresión protegible, podían o debían entrar en la fase de comparación y en definitiva considerarse parte de la expresión.

Debe recordarse que un programa *es en esencia un algoritmo* y no obstante lo anterior se halla si protegido de manera generalizada por el derecho de autor. Esto se traduce en que no es tanto que el elemento *sea o no un algoritmo* o algún método de operación, para filtrarlo sin más, siendo relevante el hecho de si hay o no expresión creativa al analizar la forma concreta en que el programador ha implementado dicho algoritmo o método. A mi juicio, sucede que en realidad los algoritmos o métodos de operación comúnmente se disponen para funcionar eficiente o sucintamente, y por esta razón, y en dichos casos, las posibilidades creativas del autor son nulas y terminan identificándose con la idea subyacente: la función misma no protegible.

---

[161] Ob. cit. supra nota [126].
[162] Ibid.

B.3.) COMPARACIÓN.

Utilizando el sustrato protegible obtenido de la fase de abstracción y filtración, el tribunal procede a comparar los programas implicados en el juicio y determinar si ha habido infracción a derechos de autor, en la forma de copia de la obra original o reproducción, pudiendo también ser utilizado para determinar si una obra derivada de otra original (Ravicher, 2002)[163].

En ese sentido, las fases de abstracción y filtración permiten hallar el sustrato protegible en los programas, su expresión original, y la comparación permite determinar si ha habido infracción de derechos de autor en el caso concreto, principalmente el derecho de reproducción (y generalmente de algún elemento no literal del código).

[163] Ob. cit. supra nota [156].

7. Derecho de Autor Chileno en los Programas: Alcance de cada Derecho.

Según Hollaar (2002)[164], producto de la complejidad y extensión de los programas, la mayoría de las infracciones a los derechos de autor se reducirá a copias textuales (*elementos literarios literales*) que no requiere de un análisis tan exhaustivo *a la* AFC, o bien, casos de distribución no autorizadas. Esto implica que, en general y por no haber cambiado las circunstancias a que alude Hollaar, la determinación de los contornos de la expresión y la susceptibilidad concreta de protección de un programa no serán aspectos debatidos comúnmente en juicio[165], cobrando relevancia el contenido de los derechos otorgados.

A diferencia del sustrato protegible en los programas o "expresión original", los contornos de los distintos derechos de autor, su contenido y limitaciones, están dispuestos mucho más claramente en la LPICh. Los diferentes derechos de autor que se otorgan ante el supuesto de una obra intelectual se traducen en un catálogo de actos que solo el titular o quienes estén expresamente autorizados pueden realizar. El estudio de los derechos de autor en los programas apunta a determinar en qué consisten dichos actos, utilizando el lenguaje de que dispone la LPICh pero adaptado debidamente al fenómeno de los programas.

Se advierte desde ya que el primer obstáculo interpretativo, ya aludido, es el concepto de "copia de programa" presente en el art. 5 letra t. El remedio general para este obstáculo, sin entrar a inventar una teoría de cosas digitales que renueve o circunvenga el sistema de propiedad civil, consiste en interpretar el concepto de "copia de programa" de manera indistinta, como el dispositivo de hardware en que se almacena un programa, o como la información en sí (el software autónomamente considerado del dispositivo físico), según cobre mayor sentido o, por último, evitando la reducción al absurdo.

7.1. Derecho de reproducción.

[164] Hollaar, L. A. (2002). *Legal Protection of Digital Information*. Obtenido de *Digital Law Online: Software Copyright:* http://digital-law-online.info/lpdi1.0/treatise17.html

[165] Este comentario debe en realidad tomarse *cum grano salis*, ya que el insigne y aún no resuelto caso *Oracle v. Google* comprende muy intensamente la problemática de los elementos literarios no literales y la determinación de la expresión original en un programa.

El artículo 5 letra u) define reproducción como “la fijación permanente o temporal de la obra en un medio que permita su comunicación o la obtención de copias de toda o parte de ella, por cualquier medio o procedimiento”.

7.1.1. Copia y guardado de archivos.

El procedimiento de fijación u obtención de copias en los programas se identifica con la duplicación de la información de que consta el programa. La información digital se organiza en archivos computacionales o *files.* Un programa puede en realidad hallarse distribuido en varios archivos distintos y hallarse en estado ejecutable (archivo digital de código objeto) o no ejecutable (archivo digital de código fuente, o de código objeto al que aún le faltan más pasos de procesamiento, ej. enlaces estáticos). Todo archivo debe residir en un dispositivo de almacenamiento físico, tal como un disco duro, un dispositivo USB flash (*pendrive*) o un CD-ROM, por nombrar los más insignes. Este es el “procedimiento” en que se fija y reproduce la información digital y el cual se subsume en el concepto legal de reproducción.

Si el archivo es copiado y guardado en un dispositivo que solo pueda contener dicha información, a la luz de nuestra LPICh estamos en justa presencia de una “copia del programa” en su forma tangible expresa. En realidad y a esta altura tecnológica, sea un dispositivo externo o el mismo disco duro, lo usual es que se un dispositivo físico determinado contenga información de distinta naturaleza, autor y fuente (ej. un pendrive con 5 programas distintos y 5 imágenes). En estos casos el concepto de “copia de programa” conduce al absurdo (la copia se identifica con todo el dispositivo contenedor que es común a varias obras) y debe conducir a la información precisa implicada.

Uno de los casos típicos en que ocurre copia y guardado de archivos es el software de descarga, especialmente en sitios web de público acceso. La descarga consiste en una técnica copia del archivo que reside en el servidor físico, a efectos de guardarlo en un medio físico del computador cliente, usualmente su disco duro. A mi juicio, y siendo discutible, el que realiza la reproducción es el cliente, ya que el proceso de descarga y todo lo que conlleva se produce únicamente a instancia de este: el dueño o arrendatario del servidor solo pone el archivo a disposición de otros, circunstancia que se subsume más, como se verá, con el concepto de distribución o comunicación pública, según se estime. Debe advertirse que esta copia realizada por el cliente es una copia de adquisición (la copia persigue obtener la información de que consta el programa) y no una de las copias a que alude el art. 71 Ñ letra a (copias

de un programa que ya se "tiene", que ya se ha adquirido, a efectos de ejecutarlo). En perspectiva del quien posea, arriende o administre el servidor y el servicio de descarga ofrecido, además de comprender su acto un supuesto de posible distribución o comunicación pública, es indudable que esta persona debió tener acceso, en algún momento, a alguna copia del programa que ofrece para descargar, sea por haberla adquirido (ej. por propia descarga) o por ser el creador original del programa o alguien autorizado por este también para su reproducción (y distribución). Se tiene que, técnicamente, en la descarga, aquellos que estén operando el cliente y el servidor, están reproduciendo la obra, pero en circunstancias y momentos distintos.

Por el alcance del derecho de reproducción en el ámbito digital, los autores pueden controlar legítimamente la cantidad de copias que puede hacer el usuario sobre el programa. Es usual que un programa solo requiera una única instalación por cada computador físico y que así sea especificado en la licencia de software. Más aún, debe advertirse que estos actos de copia se subsumen como constitutivos de reproducción con independencia de las conductas ulteriores posibles en que incurra el que obtenga dichas copias (ej. el que obtiene la copia procura a su vez borrar el archivo que le sirvió de matriz, quedando efectivamente un solo archivo, simulando una suerte de transacción de cosas materiales, caracterizada por la exclusividad) algo que se observó en el caso *Capitol Records, LLC v. ReDigi Inc.* (S.D.N.Y., 2013).

7.1.2. Carga del programa.

Otro acto constitutivo de reproducción, tal vez menos obvio, es la carga del programa. La carga es una fase esencial de todo programa que se pretenda ejecutar en un computador y consiste en la copia del programa desde el almacenamiento a la memoria (ej. *RAM*). El considerar la carga como reproducción ha sido recogido al menos desde el año 1993 en el caso *MAI Systems Corp. v. Peak Computer, Inc.* (1993, 9th Circuit, "MAI"), pese a que la jurisprudencia norteamericana tenía un obstáculo interpretativo para así estimarlo[166] al requerirse que exista permanencia en la fijación (algo cuestionable respecto a la *RAM*, la cual "pierde" la información al apagarse el computador). En todo caso, al tenor de nuestra LPICh, la

[166]En este caso, el *9th Circ.* sorteó el obstáculo implicando que el requisito de permanencia apunta a que sea (cita textual) "sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of *more than transitory duration*".

reproducción basta que sea temporal, por lo que la carga puede considerarse como una forma de reproducción.

Que la carga constituya reproducción no quiere decir que la ejecución del programa también lo sea. Sin embargo, debe observarse que la ejecución requiere perentoriamente una carga previa y que la situación de que la ejecución *sea* efectivamente otra forma de reproducción se puede hallar implícita[167], o incluso explícita[168] en algunos sistemas. Técnicamente, la ejecución no entraña copia adicional a la que si debió ocurrir en la fase de carga, sin embargo es efectivo que ejecutar un programa supone necesariamente un acto de reproducción *en algún momento*. Nuevamente y bajo similar razonamiento, puede decirse que los *displays* del programa suponen que ha habido reproducción del código, pero no entrañan una reproducción adicional del código, aunque si involucran una suerte de reproducción de archivos y otra clase de información de software tales como imágenes, videos, música, etcétera. Algunos sistemas consagran de alguna manera la circunstancia de que la obtención de resultados audiovisuales es un acto de reproducción del código[169], algo que no es sostenible en la técnica y que atenta contra la separabilidad de las obras implicadas, a la vez que no comprende que los displays del programa involucran también obras creadas por autores distintos. A efectos de nuestro sistema, es conveniente mantener la separación de ámbitos porque los displays tienen una naturaleza distinta al programa como código u *obra literaria* al tenor del art. 2 del Tratado OMPI de 1996. En contra, algunos podrían sostener que los *displays* son sino el ejercicio del derecho de ejecución o comunicación pública que opera en el código, pero nótese como estos derechos están limitados a un contexto calificado.

7.1.3. Limitante de obtención de copias de seguridad.

Aún con esta base de actos posibles, la LPICh contempla un catálogo de actos que, pudiendo subsumirse como actos constitutivos de reproducción exclusivos del titular, se encuentran permitidos o excluidos del ámbito de derecho de autor. En nuestro sistema, las limitaciones y excepciones del derecho de autor en los programas, abarcando principalmente los derechos de reproducción y adaptación, se hallan casi en su totalidad en el artículo 71 Ñ. Prácticamente todo el artículo 71 Ñ tiene como fuente mediata las

[167]Se halla implícita, como se verá, en aquellos sistemas en que la defensa a la infracción del derecho de reproducción del "uso esencial" (análogo a nuestro art. 71 Ñ) se restringe únicamente en favor de los "dueños" del programa.
[168]Se halla explícita ("running") en el artículo 4.1. letra a) de la Directiva UE (2009).
[169]Se halla explícita ("displaying") en el artículo 4.1. letra a) de la Directiva UE (2009).

recomendaciones de la CONTU referidas a los usos justos de los programas (1978) y que determinaron el texto actual de la Sección 117 del 17 U.S. Code.

El primero alude a la obtención de copias de seguridad o *backup copies*: copia y guardado a partir de una copia matriz a efectos de resguardar una eventual destrucción o inutilización de la primera, incluyendo la eventual destrucción o inutilización del computador físico (en realidad el disco duro) que la almacena. Por esta última situación, es usual que la copia se prepare y almacene en un dispositivo externo al computador principal (ej. pendrive, CD-ROM, servidores externos). En ese sentido y utilizando los conceptos de archivo y/o respaldo, el citado artículo 71 Ñ reza que: "las siguientes actividades relativas a programas computacionales están permitidas (…) a) La (…) copia de un programa computacional efectuada por su tenedor, siempre que la copia sea (…) para fines de archivo o respaldo y no se utilice para otros fines".

Sea bajo los conceptos de "copia de seguridad", "copia de archivo" o "copia de respaldo", la limitante de la obtención de copias de seguridad tiene sus contornos claramente definidos. Explicando tanto la limitante de las copias de seguridad y la defensa de uso esencial (que se verá a continuación), la CONTU (pp. 31) sostuvo que (traducción propia):

"Producto de que la introducción de un programa en un computador es la preparación de una copia, la ley debe velar que los poseedores legítimos de una copia de un programa puedan utilizarlos de manera libre y sin miedo a exponerse a demandas de derechos de autor".

Explicando la forma en que la ley debe velar tal situación, la CONTU agrega, a modo meramente ejemplar:

"El derecho a cargar el programa en un computador y el de preparar copias de archivo de este para resguardarse ante su destrucción o de daños producto de fallas mecánicas o eléctricas (…) Este derecho no se debería extender a otras copias del programa".

No es posible hacer valer esta limitante por aquel que ejecuta tanto la copia matriz como la copia obtenida de esta, de manera indistinta o en computadores distintos. Una o más copias de seguridad serán comprendidas como tales, cuando resguarden ante la contingencia de destrucción o inutilización (ej. el programa falla irremediablemente y no puede ejecutarse). Ocurrida dicha contingencia, entrará

recién en acción la copia de seguridad, pudiendo el usuario disponer la obtención de nueva o nuevas copias de seguridad a efectos de resguardar la primera, la cual pasa a ser *nueva* matriz.

7.1.3. Limitante relacionada al uso esencial del programa.

El mismo artículo 71 Ñ sostiene que: "las siguientes actividades relativas a programas computacionales están permitidas: (…) a) (…) *copia de un programa computacional* efectuada por su tenedor, siempre que la (…) copia sea *esencial para su uso* (…) no se utilice para otros fines". En cierta manera, aunque bajo idea subyacente afín o análoga, también puede citarse el art. 71 O ("reproducción provisional de una obra (…) tener como única finalidad (…) el uso lícito de una obra (…) que no tenga significación económica independiente).

El "uso esencial" de que habla nuestra LPICh en su art. 71 Ñ letra a, apunta a la ejecución del programa, no siendo relevante a estos efectos, la destinación concreta de dicha ejecución, vale decir, se trata del uso esencial del programa no del uso del programa según su esencia o naturaleza (ej. emplear un procesador de texto para escribir y no para fines lúdicos). El ejemplo paradigmático de una copia realizada para darle uso esencial al programa es precisamente la carga del programa, situación que ya advertía y comprendía la misma CONTU hacia 1978. Respecto de la ejecución del programa así como la obtención de resultados audiovisuales, de estimarse como reproducciones (discutible, véase arriba), debe decirse que también entran, con mayor razón, como uso esencial. Podría argumentarse que la obtención de los resultados audiovisuales constituye una adaptación del código y por ende del programa, tal como una dramatización constituye una adaptación de un guion teatral, pero, aún en esta interpretación, se debe tener en cuenta que la defensa del uso esencial en nuestra LPICh recibe aplicación también a propósito de las *adaptaciones*, por lo que se está a la conclusión precedente.

No se hace válida por la vía del uso esencial la obtención de copias para ser instaladas y/o almacenadas *en otros* computadores, sin perjuicio de aplicarse la limitante de copias de seguridad cuando corresponda y el contenido de la licencia a este respecto. Esta conclusión es independiente del importante concepto de "tenedor" que emplea la disposición analizada, ya que el acto en sí no busca ejecutar el programa, sino duplicar la información para otros fines (ej. copiar el programa ya instalado para almacenarlo y grabarlo en un CD-ROM, todo a efectos de utilizarlo *en otros computadores*).

El caso de la obtención de copias ilegales o "piratas" en nuestra LPICh es quizás curioso. Como los desarrolladores pueden controlar la obtención de copias del su programa, aquellos que no se amparen

en los supuestos de copia de seguridad o uso esencial, están efectivamente infringiendo el derecho de reproducción, utilicen o no dispositivos externos en el procedimiento de duplicado de información y con independencia de las cláusulas contractuales que gobiernen el uso del programa (*covenants* en la licencia). Sin perjuicio de esto, la ley les dispensa de infringir *nuevamente* el derecho de reproducción al cargar y ejecutar la copia infractora obtenida, ya que le basta que un "tenedor" realice dichos actos, sin perjuicio de que este último concepto, propio del sistema de propiedad civil de nuestro Código, es problemático de aplicar a la información. Si la ley hubiese contemplado un concepto diverso, por ejemplo, "adquiriente legal" o "tenedor legítimo", o limitando la extensión de manera objetiva, vale decir, aludiendo a copias *autorizadas*, en la misma situación anteriormente descrita, la aptitud infractora de un usuario que realiza copias piratas y posteriormente las emplea en un computador, hubiese sido, al menos, doble: se infringe el derecho de reproducción al realizar la copia, y se infringiría nuevamente al cargar y ejecutar la copia obtenida, por constituir la carga o ejecución una copia en sí y por no ser el infractor un titular de la defensa del uso esencial.

El concepto de *tenedor* en la LPICh, la situación jurídica que se debe tener para hacer valer la limitante del uso esencial, es más menos trivial a efectos de determinar el reproche jurídico sobre los actos que encaminados a obtener copias ilegales de programas (hacer la copia ya es una infracción), por la extensión erga omnes del derecho de reproducción, pero la situación jurídica del que puede hacer valer la defensa del uso esencial si es relevante en otros aspectos. En el sistema norteamericano, la defensa del uso esencial solo puede hacerse valer por aquel que se entiende como "dueño" (*owner*) del programa, concepto igual de problemático al utilizarse para referir a bloques de información inmaterial. Aún con esta falte de "calce perfecto" y, tal como se pudo observar en el Capítulo 4, la jurisprudencia norteamericana ha aplicado la situación jurídica de dueño en la información por vía de analogía.

La jurisprudencia norteamericana aborda implícita o explícitamente el problema de las situaciones jurídicas que puede tener una persona respecto de determinado bloque de información, sin necesidad de crear una nueva categoría de cosas, pero que aplicada a la defensa del uso esencial, se vuelve peligrosa en un escenario de comercialización de software en que una de sus expresiones particulares, son los acuerdos de licencia para el usuario final o *EULAs* caracterizados por las intensas y hasta desconcertantes restricciones que imponen a los usuarios. Siguiendo la analogía, sobre dichas transacciones en base a *EULAs* se vuelve tentador considerar la situación jurídica de los usuarios como

análoga al de un *mero tenedor*, en el sentido que debe descartarse cualquier situación de propiedad o calidad de *dueño,* ya que las restricciones efectivamente impuestas a los usuarios nacidas del contrato de licencia apuntan a que tienen sino una situación precaria, transitoria, y muy limitada. Las cláusulas presentes en algunos contratos de licencia en que se expresa que el programa es licenciado y no vendido ("licensed not sold")[171], y por ende no siendo un título de propiedad o de alguna situación jurídica remotamente relacionada a la propiedad, se agregan como argumentos en ese mismo sentido.

Bajo este escenario interpretativo, que puede catalogarse como regresivo, para asentada jurisprudencia norteamericana, actos como la carga del programa, que entran en el concepto de reproducción previsto en el sistema derecho de autor y que solo se excluyen de dicho ámbito únicamente para los *dueños*, dejan entonces de ser admitidos como *fair use*, precarizando, sin razón, la situación de los usuarios de programas, efectivamente "anulando" la aplicación de la Sección 117 (Hollaar, 2002)[172] que es el análogo a nuestro artículo 71 letra Ñ, así como los aportes de la CONTU que buscaban fundamentar el interés de los usuarios en la protección jurídica de los programas por el derecho de autor[173]|[174]. Esta exacta circunstancia, fue llevada a la práctica jurisprudencial de dicho país con la decisión de la Corte de Apelaciones para el *9th Circ.* en el caso *MAI* referido[175] y que ha sido criticado por jurisprudencia norteamericana posterior, e incluso por el mismo *9th Circ*. Después del caso *MAI*, en el caso *Blizzard*, el tribunal de la causa reconoce las posibles consecuencias económicas y jurídicas de la interpretación que precariza la situación de los usuarios, para evitar una regresión jurídica y un atentado a las ideas que informaron la protección de los derechos de autor en los programas, niega subsumir los actos que

[171]Nótese que si se interpreta las cláusulas *licensed-not-sold* presentes en los contratos de licencia como referidas a derechos de propiedad intelectual estas carecen de sentido jurídico alguno, en lo que a los intereses de los desarrolladores se refiere. ¿Por qué razón el desarrollador estaría haciendo hincapié que no está "vendiendo" sus derechos de propiedad intelectual sobre el programa? La razón es que el desarrollador se está refiriendo a la información en sí, está siendo el objeto de la licencia de software. Si uno examina tratando de hacer sentido práctico a las diversas licencias de software propietario o privativo notará que los derechos de propiedad intelectual se reservan celosamente, salvo casos concretos (ej. es usual que los *freeware* admitan libre distribución), por lo que no hay propiamente una licencia de derechos de autor.

[172] Ob. cit. supra nota [164].

[173]Al menos al año 1998, el Congreso de Estados Unidos ya comprendía esta situación de inaplicabilidad de facto que tenía la Sección 117 y reaccionó mediante una enmienda dicha sección dada por la *Computer Maintenance Competition Assurance Act* (1998), disposición que, en lo fundamental, incluía nuevas limitantes al derecho de reproducción en los programas pero con extensión y supuestos limitados a la mantención y reparación de computadores. Resulta curioso el hecho de que en el texto de la nueva letra c de la Sección 117 se utilice el concepto de "copia autorizada" y no utilice los incómodos conceptos de "dueño", "tenedor" o poseedor".

[174]Aquí aludo a las recomendaciones concretas de la CONTU en la materia (pp. 30-33).

[175]La denegación de la *essential-step defense* basada en la dualidad dueño-licenciatario sucedió igualmente en el caso *Vernor* comentado en el Capítulo 4.

tienen ocasión en la ejecución del programa como actos propios de derecho de autor, pese a que también deniega la aplicación de la limitante del uso esencial (la licencia no es un título de "propiedad" sobre el programa, luego el usuario no es un dueño y se descarta la aplicación de la Sección 117). El tribunal comenta así que "Blizzard – u otro desarrollador de software protegido- puede sancionar cualquier conducta realizada durante la ejecución del software como infracción a sus derechos de autor (…) la razón sería que, por cuanto la conducta ocurre mientras el computador del usuario está *copiando* el código del software en la RAM para ejecutarlo, la conducta involucra infracción de derecho de autor. *Esto otorgaría a los desarrolladores de software protegido muchos más derechos de los que el Congreso ha generalmente otorgado a los titulares de derecho de autor*".

En nuestro sistema, aplicando el mismo razonamiento analógico presente en *Vernor* y abandonando la creación jurisprudencial de una nueva categoría de cosas digitales, aún quienes tengan interés restringido o una situación jurídica precaria sobre el programa son candidatos para hacer valer la defensa del uso esencial, de modo que el interés de un usuario de programas no se compromete inútilmente. Tal como reconocía la CONTU, resulta bastante poco probable que un desarrollador quisiera demandar por vulneración a sus derechos de autor a un usuario adquiriente legítimo de su programa. Los desarrolladores buscan precisamente que sus programas sean ejecutados y la defensa del uso esencial busca principalmente excluir la ejecución del programa, al menos del ámbito de derecho de autor. En ese sentido, bajo nuestra LPICh no hay necesidad de entrar a crear jurisprudencia de "última hora" para defender a los usuarios que realizan actos cuya aptitud para infringir derechos de autor es cuestionable o inexistente.

Amparado en nuestro art. 71 Ñ letra a y en normas comparadas similares, cualquier usuario puede hacer valer la defensa del uso esencial. Por esta razón, no se comparte en este ensayo la interpretación de que si un programa es distribuido a un usuario sin *licencia de derecho de autor* alguna, este quedará según el programador Atwood (2007)[177] "sin derecho legal para *utilizarlo*", un comentario bastante generalizado en el entorno en cuestión. Tal situación es incluso paradójica, ya que distribuir un programa al público, especialmente el código fuente, sin acompañar licencias de derecho de autor, generalmente no tiene como asidero en venir a *restringir* la utilización del programa, sino todo lo

[177] Atwood, J. (4 de abril de 2007). *Pick a License, Any License*. Obtenido de *Coding Horror:* https://blog.codinghorror.com/pick-a-license-any-license/

contrario. Es cierto que el usuario no podrá utilizar el programa en ninguna de las formas que define el artículo 18 de la LPICh, pero utilizar el programa en su sentido obvio, vale decir, ejecutarlo con un computador y emplear los resultados de cómputo para los fines que estime, puede hacerlo sin comprometer su responsabilidad por la vulneración de derechos del autor del programa en cuestión, debido a la extensión y alcance del concepto de tenedor en el art. 71 Ñ letra a. Con todo, la delimitación más precisa de lo sostenido dependerá grandemente del carácter normativo que se le otorgue a la limitación de la letra a del art. 71 Ñ[178].

Con todo, el concepto de "tenedor" utilizado en nuestra LPICh presenta algunos aspectos problemáticos. Siguiendo tangencialmente a Hollaar (2002)[179], el primero de estos radica en un caso puntual relacionado al arrendamiento de copias del programa, acto relacionado a nuestro art. 37 bis de la LPICh. Sin perjuicio de que el arrendamiento de copias físicas de programas es parte del otrora relevante software de retail, piénsese en el caso de una persona que arrienda el ámbito físico y que luego realiza una o más copias de respaldo en su disco duro basándose en el art. 71 Ñ letra a. Aún después de verse en la necesidad de devolver la copia física, es posible continuar la ejecución del programa de manera totalmente válida a la luz de la LPICh: carga o uso esencial de copias de respaldo que se obtuvieron previamente, observando las restricciones a la transferencia de dichas copias de respaldo al tenor de este y que se verán más adelante. Esto se debe al hecho compuesto de que, por una parte, en concepto de nuestra ley, toda clase de tenedor (con o sin título) es candidato para la defensa del uso esencial y, por la otra, nuestra ley solo contempla debidas limitaciones al derecho de distribución a propósito de las copias de respaldo. Si esta situación se generaliza, situación poco probable por la baja extensión del arrendamiento de copias físicas de programas, el legislador nacional puede remediarla, incluyendo la obligación expresa de destruir las copias de respaldo obtenidas una vez regresada que sea la matriz arrendada, o incluyendo el adjetivo de "legítimo" al de tenedor u otro concepto similar (ej. copias autorizadas; copias legalmente detentadas). Lo que no debe hacer,

[178] Este es un problema de territorialidad, pero también del carácter o naturaleza de las normas de excepciones o limitaciones del sistema de derecho de autor. Al menos respecto del art. 71 Ñ en su totalidad, pareciera que se está consagrando un auténtico "derecho del usuario de programas" y no una delimitación del alcance efectivo del derecho de autor de reproducción y de adaptación. Esta interpretación, si bien fundada razonablemente en el lenguaje utilizado, puede producir roces entre los sistemas internos de derecho de autor implicados, vale decir, el derecho aplicable.
[179] Ob. cit. supra nota [164].

aprendiendo del error norteamericano, es cambiar el concepto de tenedor por el de dueño o el de poseedor, porque en este caso, la balanza se irá muy en perjuicio de los usuarios.

El segundo aspecto dudoso consiste en que el concepto de "tenedor" deja en absoluto pie de igualdad a quienes obtienen copias por canales autorizados de los que no, salvo que estimemos, necesariamente, que actos como la descarga son formas de reproducción según nuestra LPICh. En ese sentido y salvo que admitamos lo anterior, puede decirse que obtener una copia de un programa por medios o canales no autorizados (ej. una página web sostenida por persona distinta al desarrollador y sin licencia competente para distribuir) no es una infracción a los derechos de autor, sin perjuicio de la clara infracción del distribuidor. Utilizando los mismos artículos de la LPICh mencionados, respecto del software contenido en hardware contenedor así distribuidos ("CDs piratas"), habría que necesariamente distinguir si el que adquiere la copia es la misma persona que realiza la reproducción no autorizada y distribución[180], en su caso. Es curioso que la LPICh no contemple presunciones legales al respecto (ej. presumir que el detentador de una copia ilegal fue el que la produjo).

Teniendo en cuenta todo lo anterior, un aspecto que podría haber consagrado nuestra ley consiste en limitar la defensa de uso esencial y de copias de seguridad solo respecto de *copias obtenidas legalmente*, aquellas adquiridas a través de medios y canales autorizados por el desarrollador. Inexplicablemente, este tratamiento discriminatorio pero no arbitrario, si está previsto a propósito de las siguientes limitantes.

7.1.5. Limitantes relativas a la ingeniería inversa.

El artículo 71 Ñ letra dispone en su letra b:

"Las siguientes actividades relativas a programas computacionales están permitidas (…) b) las actividades de ingeniería inversa sobre una *copia obtenida legalmente* de un programa computacional que se realicen en el único propósito de lograr la compatibilidad operativa entre programas computacionales o para fines de investigación y desarrollo. La información así obtenida no podrá

[180]Esto porque *adquirir* la copia (ej. comprarla en comercio no autorizado) no se enmarca en ni un acto de reproducción ni de distribución. Tanto este adquirente como los adquirentes de canales autorizados podrán ejecutar el programa sin violar derecho de autor alguno, al ampararse en el uso esencial (ej. ambos son "tenedores"). Es en la descarga donde se produce la curiosa situación (véase supra) en que la adquisición es, técnicamente, una copia.

utilizarse para producir o comercializar un programa computacional similar que atente contra la presente ley o para cualquier otro acto que infrinja los derechos de autor".

Luego, la letra c del mismo artículo sostiene: "las actividades relativas a programas computacionales están permitidas (…) c) las actividades que se realicen sobre una copia obtenida legalmente de un programa computacional, con el único propósito de probar, investigar (…) su funcionamiento".

Enmarcada en el ámbito de los programas, la ingeniería inversa puede definirse como un conjunto de operaciones realizadas sobre un programa, generalmente como producto final o terminado, encaminadas a determinar los detalles de su funcionamiento. Tal como notaban Cid & Merello (2007)[181], el desarrollo o ingeniería de software parte de las funciones o necesidades de la demanda y desemboca en un código terminado. La ingeniería *inversa* pretende inferir los detalles de las funciones (ej. estructura, relaciones presentes entre cada módulo) a partir del código, el cual no es siempre accesible en la forma de código fuente comprensible. Más concretamente, las "actividades de ingeniería inversa" aludidas por la LPICh consistirán en la examinación u observación del programa en ejecución, resultados audiovisuales inclusive, y la descompilación del código (traducción del código objeto ininteligible a código fuente en lenguaje de alta abstracción compresible). De esta manera, el propósito más usual de la ingeniería inversa en los programas trata sobre la reconstrucción del código fuente fácilmente entendible (a veces no disponible) a partir del inescrutable código objeto (si disponible), operación que es denominada por algunos como desensamblaje o *disassembly* (Hollaar, 2002)[182][183]. Debe recordarse que nuestro sistema protege al código fuente y objeto por igual, aunque en caso de que se acompañe el código fuente (ej. FOSS), la ingeniería inversa no tiene real razón de ser.

Como solo los titulares de la obra tienen el derecho exclusivo de distribuir la obra, la obtención del código fuente u objeto por medios no autorizados, al tenor de los requisitos de la letra b, no permite dar aplicación a esta limitante: se requiere obtener una copia por medios legítimos, esto es, se requiere, en contraste a la defensa de la copia de seguridad y del uso esencial, ser un "tenedor legítimo". ¿Cómo se define el concepto de tenedor legítimo en un programa teniendo en cuenta la impropiedad

[181] Cid, J., & Merello, A. (2007). *Patentes de Invención y Software.* Santiago. pp. 59-63.
[182] Ob. cit. supra nota [164].
[183] Si bien Hollaar no lo precisa, sucede que el *desensamblaje* es un concepto técnico en la ingeniería de software referido a la traducción del código en lenguaje de la máquina a código en lenguaje ensamblador, este último caracterizado por ser de baja abstracción y que requiere conocimientos moderados de los detalles del computador físico. En ese sentido, es mejor simplemente hablar de *descompilación*.

conceptual? Podría decirse que identifica a toda aquella persona, natural o jurídica, que ha obtenido la información de software, sin importar la intensidad de las restricciones que se le apliquen, a partir de distribuidores autorizados, sean de canales físicos o digitales, o que de alguna otra manera cuenta con la aquiescencia del autor para emplear y ejecutar el programa.

Volviendo a naturaleza de los actos constitutivos de ingeniería inversa, podemos desde ya identificar que las actividades relacionadas a la traducción del código objeto a código fuente (ej. descompilación) se relacionan, al menos a priori, a actos constitutivos del derecho de adaptación. Es conveniente que el concepto técnico de "traducir un programa" pueda subsumirse con facilidad en los arts. 5 letra w y 18 letra c de la LPICh, que aluden a la traducción en su acepción natural. Con todo, los detalles del derecho de adaptación se verán más adelante.

La razón por la cual nuestro legislador ha decidido permitir los actos de ingeniería inversa, se puede relacionar con en el artículo 71 B que dispone la limitante de derecho de autor aplicada a toda obra y que refiere concretamente a la "inclusión en una obra (…) que haya sido lícitamente divulgada, y su inclusión se realice (…) con fines de (…) enseñanza e investigación". También es necesario traer a colación el ya analizado art. 2 del Tratado OMPI de 1996.

Además del fin investigativo, inherente en toda actividad de ingeniería inversa, y que se redunda posteriormente por el propio art. 71 Ñ en su letra c, nuestra ley agrega el *logro de interoperabilidad funcional* programa a programa *o* "software-software" como razón plausible para entrar a realizar actividades de ingeniería inversa, situación que entraña otra redundancia. La interoperabilidad, habíamos dicho al principio del presente ensayo, se expresa mediante el diseño, adopción y logro de estándares e interfaces, todo lo cual se dispone a efectos de lograr optimizar las actividades de procesamiento de la información. La efectiva adopción de estándares e interfaces requiere, desde luego, conocer los detalles del funcionamiento de los distintos componentes en juego (primer paso, conocer las piezas), pero su centro real está en la modificación de estos componentes a efectos de lograr la optimización o interoperabilidad misma (segundo paso, unirlas efectivamente). Debe tenerse en cuenta que la interoperabilidad software-software (ej. aplicaciones con el sistema operativo), a que parece limitarse la disposición en comento, es una de las tantas expresiones que tiene el principio de interoperabilidad, por lo que, aparentemente, no todo acto encaminado a lograr necesaria interoperabilidad se alza como motivo eligible para fundar la defensa de la ingeniería inversa. Sucede

en realidad que la otra expresiones de interoperabilidad está desparramadas en el propio art. 71 Ñ. El logro de interoperabilidad software-hardware como limitante, está implícito en la letra a del mismo artículo, disposición ya visitada: "adaptación o copia (…) siempre que (…) sea esencial para su uso", debiendo agregarse al final la obvia y en realidad redundante frase "con un computador", y también en la letra c: "actividades (…) con el único propósito de probar, *investigar o corregir su funcionamiento* o la seguridad (…) *de la red o del computador* sobre el que se aplica". Generalmente, el logro de interoperabilidad se logra mediante actos concretos relacionados a la modificación de un programa y por ende más relacionados al derecho de adaptación. Así, un mayor análisis de la interoperabilidad se difiere para dicho derecho.

Una razón más fundada, y que explica la inclusión especial y más detallada de la limitante de derecho de autor de usos que persigan fines meramente investigativo, siempre a propósito de los programas, el cual había sostenido que en su grueso fue redundado por el art. 71 Ñ letra b, refiere a, siguiendo la jurisprudencia norteamericana en el caso *Sega Enterprises Ltd. v. Accolade, Inc.* (1992), que si los actos de ingeniería inversa fueran "per se vulneradores de derecho de autor", tomando en cuenta que los programas usualmente se distribuyen solo en código objeto imposible de entender por humanos y por potenciales interesados en investigar el funcionamiento del programa, es que se estaría confiriéndoseles a los desarrolladores de programas un "monopolio de facto sobre las funciones" (el objetivo real de la ingeniería inversa), circunstancia extraña al sistema de derecho de autor y más propia de un derecho de patente o un secreto empresarial.

¿Cuál es la relevancia de la admisión de la ingeniería inversa en sede de derecho de autor con respecto del sistema de patentes? Depende de la forma y alance en que el ordenamiento recoja la protección de programas. Según Hollaar (2002)[184], una patente tiene relación con las funciones y características inventivas que puedan existir a propósito de un programa, y puede reclamarse sobre estos o como un método (los algoritmos en sí[185]) o como una máquina, en este último caso, vale decir, un computador físico con el programa ejecutándose. Más que aludir a las actos de ingeniería inversa en sí, suponiendo que existe una patente a propósito de un programa, el derecho de patente vigente impide que terceros realicen los actos que prescribe el art. 49 de la Ley 19.039, fundamentalmente la explotación comercial

[184] Ob. cit. supra nota [164].
[185] Se clarifica que se alude a un concepto amplio de algoritmo no delimitado a los puramente matemáticos o lógico-formales.

de los procesos inventivos subyacentes en el programa, teniéndose en cuenta que es ineludible en la solicitud de la patente el acompañar una descripción del invento (arts. 43 y 43 bis inciso final de la Ley 19.039) que permita reproducir la invención sin más antecedentes. Esto último implica que los actos de ingeniería inversa carecen de real sentido práctico, ya que los detalles del funcionamiento del programa serían públicos. Quién obtiene legítimamente un programa puede analizarlo, investigarlo y así descubrir los distintos detalles de su funcionamiento, pero, en caso de que exista efectivamente una patente, no por poder realizar las investigaciones pertinentes podrá emplear las funciones, procesos y métodos patentados (Menell, Lemley, & Merges, 2019)[186]. Respecto de los secretos comerciales potencialmente aplicables a un programa, y teniendo en cuenta la jurisprudencia y doctrina norteamericana[187], las actividades de ingeniería inversa son maneras "legítimas"[188] de obtener el secreto, ya que se relacionan a la investigación y desarrollo independientes (Menell, Lemley, & Merges, 2019)[189].

La admisión de la ingeniería inversa contempla eso si una advertencia, con un alcance quizás sobrevalorado:

"La información así obtenida no podrá utilizarse para producir o comercializar un programa computacional similar que atente contra la presente ley o para cualquier otro acto que infrinja los derechos de autor".

En realidad, sucede que esta disposición repite que, sin perjuicio de que las funciones escapan del sistema, la expresión siempre continúa bajo tutela, de ahí que, aún aquellos que se valgan de ingeniería inversa para determinar los detalles de las funciones de un programa, no pueden pretender utilizar el resultado de dichas actividades y el conocimiento obtenido, para realizar un programa con idéntica expresión al programa desmontado, esto es, copia del código u otras obras protegidas[191], y sin perjuicio

[186] Menell, P. S., Lemley, M. A., & Merges, R. P. (2019). *Intellectual Property in the New Technological Age: 2019.* Clause 8 Publishing, pp. 88-96.

[187] *K & G Tool & Servicr Co. v. G. & G Fishing Tool Service* (Tex., 1958); *E.I. duPont de Nemours v. Christopher* (5th Circ., 1970); *Integrated Cash Management Services, Inc. v. Digital Transactions, Inc*. (2nd Circ., 1990); *Kadant, Inc. v. Seeley Machine, Inc.* (N.D.N.Y., 2003); y los *Comentarios a la Sección 1 de la Uniform Trade Secrets Act.*

[188] Art. 87 de la Ley 19.039.

[189] Ob. cit. supra nota [186].

[191] Es importante que la letra b sostenga que la ingeniería inversa como limitante, tiene como propia limitante los derechos de autor relacionados al programa en su acepción legal (código), pero también limita, en general, respecto de "cualquier otro acto que infrinja los derechos de autor". En ese sentido, la ingeniería inversa tampoco puede ser pretexto para copiar, adaptar y distribuir otras obras presentes *a propósito* de un programa (ej. composiciones musicales, imágenes, arte 3D, etcétera).

de proceder la protección de elementos no literales. No es una prohibición de ingresar al mismo mercado y alzarse como legítimo competidor (ej. hacer ingeniería inversa sobre un videojuego y luego sacar al mercado un videojuego con distinta expresión), siendo conveniente citar la misma jurisprudencia presente en *Sega* y también la jurisprudencia del caso *Atari Games Corp. v. Nintendo of America Inc.* (1992) limitándonos en este último caso al pronunciamiento del tribunal sobre la admisibilidad de la ingeniería inversa como uso justo o *fair use* a la luz del 17 U.S. Code.

En sede particular de derecho de reproducción, sin perjuicio de lo que se sostendrá a propósito de la traducción del código más adelante, la disposición de la letra b se relaciona con los actos de examinación u observación, muy especialmente a la observación de los resultados audiovisuales, los cuales requieren ejecución, y por ende carga (acto de reproducción), acto que en realidad ya se encontraba excluido del sistema según la limitante del uso esencial. En realidad, producto de que la limitante de la ingeniería inversa prescribe la necesaria adquisición legítima del programa, esta disposición viene a crear una desarmonía sistemática con la disposición de la letra a del art. 71 Ñ, la cual es conferida a cualquier *tenedor* y no solo a los legítimos tenedores.

7.2. Derecho de distribución.

El artículo 5 letra q de la LPICh define distribución como "la puesta a disposición del público del original o copias tangibles de la obra mediante su venta o de cualquier otra de transferencia de la propiedad o posesión del original o de la copia". Por su parte, el artículo 18 letra e sostiene que:

"Solo el titular del derecho de autor o quienes estuvieren expresamente autorizados por el, tendrán el derecho de (…) la distribución al público mediante venta, o cualquier otra transferencia de propiedad del original o de los ejemplares de su obra que no hayan sido objeto de una venta u otra transferencia de propiedad autorizada por él o de conformidad a esta ley".

Por último, nuestra LPICh comprende en el art. 37 bis, el derecho de autorizar o prohibir el arrendamiento comercial al público de las copias de programas o *rental of copies*[192]. Es de hecho posible encontrar sistemas que comprenden el *rental of copies* como otra expresión del derecho de distribución

[192] El Acuerdo de los ADPIC de la OMC (1994) reconoce también el *rental of copies* a propósito de obras cinematográficas. El art. 3 letra c) de la Ley 20.243 reconoce el derecho (irrenunciable e inalienable) de percibir remuneración por el arrendamiento al público en favor de artistas, intérpretes, ejecutantes y directores (según la modificación de la Ley 20.959) de obras audiovisuales (como por ej. las *screen displays* del programa).

(ej. Directiva UE 2009). Nuestro legislador comprende este derecho de manera separada al de distribución, principalmente porque el concepto legal de arrendamiento (art. 1915 del Código Civil) no puede comprenderse con la noción de "transferencia de *propiedad o posesión*" que dispone el art.5 letra q o el art. 18 letra e. Se advierte que no es posible ver en el art. 37 bis un reconocimiento implícito de la licencia de software, en tanto el objeto jurídico y económico del arrendamiento es una cosa, y el objeto económico de la licencia es la información, luego, el objeto jurídico de esta lo podríamos delimitar como un catálogo de actos permitidos o prohibidos a propósito del programa. Nótese también que el arriendo debe ser comercial, circunstancia que a mi juicio nos obliga a traer a colación el insigne artículo 3 en sus N° 1 y N°3 del Código de Comercio. En ese sentido, los arriendos y préstamos de uso meramente civiles *de contenedores de programas*, no son actos que se subsuman como una distribución, sea al tenor del art. 5 letra q o 18 letra e, o del art. 37 bis. Con todo, adviértase lo que sigue.

7.2.1. Difusión de información contenida en ámbitos tangibles.

El derecho de distribución en nuestra LPICh puede relacionarse a la comercialización de *obras protegidas*. El objeto del derecho de distribución no refiere a los *derechos de autor* del titular, los cuales se "transan" mediante competentes autorizaciones que no impliquen modificación de titular (licencias) o derechamente cediéndose. Tanto el otorgamiento de autorizaciones como las cesiones son facultades que consagra el art. 17 de la LPICh.

El concepto de distribución de la LPICh resulta problemático de aplicar al tráfico general de programas, sea comercial o no, por tres motivos: a) la comercialización de software también comprende un aspecto "intangible" (información per se), que es de hecho el centro real de la industria; b) Las situaciones jurídicas de dueño, poseedor, tenedor, así como las figuras contractuales relacionadas (ej. "ventas", "arriendos") no resultan aplicables a este ámbito propio de los programas, ya que el contrato en cuestión tiene como objeto la divulgación de información en que consiste el programa y bajo el concepto de "licencia". Sobre estos dos problemas detectados agregamos el concepto legal, incómodo en la misma forma, de "copia de programa", circunstancia ya explicitada tantas veces.

La primera solución consiste en interpretar las disposiciones legales de manera que el objeto del derecho de distribución alcance a la información y los actos y situaciones jurídicas propias y

aplicables[193]. De lo contrario, las disposiciones atingentes de la LPICh no comprenderán el supuesto ampliamente extendido del *software de descarga*, el cual quedaría legalmente excluido o estructurado de manera extremadamente artificiosa, al menos si se intenta encuadrar como distribución al tenor de los arts. 5 letra q y 18 letra e. Respecto del software de retail, a primera vista, pareciera que nuestra ley provee de un bagaje legal suficiente y bastante claro para que se subsuma como acto de distribución y no como otra cosa. En el SaaS, salvo que incida un programa de plataforma que sirva de intermediario para dar acceso al servidor, no entraña programa alguno en favor del usuario, este solo accede a servicios concretos, aunque puede decirse que existe una obra implicada de diferente naturaleza al programa como código (displays).

Comprender el fenómeno de la comercialización de software equivale a comprender el problema de la difusión de información, en este caso, de programas. También es cierto que algunas expresiones de comercialización no tienen por objeto comercial la información, sino derechamente servicios (ej. Saas). Coherentemente con lo que se ha sostenido, uno de los centros de la comercialización de software consiste en *divulgar o dar acceso* a la información en que de hecho consiste un programa, no cobrando relevancia a estos efectos que se cobre o no un precio implícito o explícito, sea único o en intervalos o si hay dispositivos tangibles adicionales como un CD-ROM. A esta altura, es muy usual que los usuarios obtengan la información de software sin pagar precio alguno y que dicha información se obtenga sin necesidad de dispositivos adicionales o concurrir a tiendas físicas.

El caso del SaaS merece una mención aparte: el usuario no obtiene información de software alguna (en ciertos casos se obtiene un programa que funciona como plataforma entre el servidor y el computador del usuario, este programa de plataforma es efectivamente licenciado, no así los servicios), y accede directamente a servicios concretos (ej. almacenamiento de archivos, edición de imágenes). Sobre el SaaS puede decirse que, sin perjuicio de que no hay "distribución" del programa como obra protegida (ej. información, código, ámbitos tangibles), consta que solo el autor del programa o quienes estén autorizados por este, pueden "prestar" un programa como servicio: se requiere de manera ineludible, tal como en el software de descarga, contar con una copia del programa a efectos de instalarla en el servidor físico, esto es, el computador que da acceso a los servicios al público general. En el SaaS, a esta

[193]Por ejemplo, la distribución aplica tanto respecto del "original" como "de las copias". La obra original en los programas apunta a la definición legal de programa que alude a las instrucciones (información) y no al ámbito tangible.

reproducción siempre necesaria le sigue la transmisión de resultados de cómputo o ejecución del programa de manera directa pero remota y no hay acceso a la información, vale decir, el usuario no tiene programa alguno almacenado o instalado en su computador. Si hemos de considerar que puede haber aplicación necesaria del derecho de autor en el SaaS, estimo que no sería el programa, en el concepto legal de obra literaria, la obra realmente implicada, sino los displays. También es posible implicar en el SaaS, por ser una especie de *streaming, ciertamente no de contenido digital, pero si* de funciones de cómputo, al derecho de ejecución pública del programa (más discutible)[194].

Por la naturaleza intrínseca de los programas como información y de la forma que en esta se "intercambia", y si no es posible sortear los obstáculos que incorpora la propia LPICh, el derecho de distribución puede terminar confundiéndose con el derecho de comunicación al público. Esto es especialmente cierto del software de descarga, en donde no hay copias tangibles, salvo que estimemos que la copia tangible implicada es el disco duro del servidor[195]. También es esperable que el derecho de distribución quede relegado conforme el SaaS siga adquiriendo fuerza (Purewal, 2012), aunque, si bien los usuarios están cada vez menos interesados en obtener la información del programa y derechamente obtener resultados y servicios relacionados, estimo que los desarrolladores, por otra parte, siempre tendrán alguna clase de interés en obtener la información. Adelantándome un poco, nuestra LPICh define comunicación pública como un "acto ejecutado por cualquier medio o procedimiento que sirva para difundir los signos, palabras, los sonidos o las imágenes, actualmente conocido o que se conozca en el futuro, por el cual *una pluralidad de personas (…) pueda tener acceso a la obra sin distribución de ejemplares* a cada una de ellas".

Como la difusión de una obra está consagrada en nuestro sistema de derecho de autor, con la presencia de copias físicas (acto de distribución o arrendamiento comercial) o sin copia física alguna (comunicación pública), el interés jurídico de desarrolladores y programadores se encuentra debidamente protegido, sea que prefieran distribuir el programa (en sentido económico y no de la LPICh desde luego) por canales físicos o digitales. Sin perjuicio de esto, la situación e intereses de los usuarios

[194]Cuando el usuario-cliente accede al servidor y utiliza los recursos de cómputo este, la ejecución efectiva del programa no se hace en un contexto realmente público, ya que es visible para cada cliente que ha accedido, de manera independiente: cada usuario-cliente accede a una instancia particular. Con todo, nuestra LPICh contempla un concepto de "contexto público" suficientemente adaptado al entorno digital caracterizado por la abstracción del concepto de "lugar físico".

[195] La LPICh es extremadamente clara al definir "copia de programa". Con todo, véase supra nota *188* para un posible argumento que permite comprender la distribución digital como auténtica distribución al tenor de la LPICh, con todas las consecuencias jurídicas que esto trae.

y del público pueden hallarse comprometidos, ya que en muchos sistemas, el nuestro inclusive, ciertos derechos presentan límites diferenciados.

7.2.2. Distribución de obras obtenidas conforme al art. 71 Ñ letra a).

El art. 71 Ñ letra a), esta vez en su parte final, sostiene que la distribución de las reproducciones y adaptaciones constitutivas de uso esencial y copia de seguridad (véase arriba) realizadas por el tenedor sobre el programa, se hallan limitadas. Como el uso esencial y la obtención de copias de seguridad son limitaciones al derecho de autor sobre el programa, estas limitaciones a la distribución (limitaciones de las limitaciones) van en claro interés del autor.

Se distinguen entonces dos limitantes a la distribución de copias de programas así obtenidas conforme a las limitantes de la letra a:

a) Las copias exactas o reproducciones "no podrán ser transferidas bajo ningún título, salvo que lo sean conjuntamente con el programa computacional que les sirvió de matriz".

b) Las copias modificadas u obras derivadas del programa "no podrán ser transferidas bajo ningún título, sin que medie autorización previa del titular del derecho de autor respectivo". Léase "titular de derecho de autor respectivo" como el titular de los derechos de autor del programa original que ha sido modificado.

Detallando un poco la razón de estas limitaciones, que esta vez operan con respecto a los usuarios, la CONTU (1978)[196] sostuvo, refiriéndose a las reproducciones obtenidas mediante uso esencial y copia de seguridad (traducción propia), que:

"Este permiso no se puede extender a otras copias del programa. Así, uno no podría, por ejemplo, preparar copias de respaldo del programa para vender algunas y retener otras para seguir utilizándolas. La venta de una copia de un programa hecha por el legítimo poseedor debe serlo de todos los derechos del programa, creando así un nuevo legítimo poseedor y destruyendo dicha calidad respecto del vendedor".

[196] National Commission on New Technological Uses of Copyrighted Works. (1978). *Final Report of the National Commission on New Technological Uses of Copyrighted Works.* Washington D.C, pp. 31-33.

Respecto de las adaptaciones u obras derivadas amparadas por el uso esencial o de seguridad, debe notarse una mayor amplitud de las limitaciones a su distribución. En efecto, la CONTU comprendió que (traducción y cursiva propias)[197]:

"A diferencia de la obtención de copias exactas descritas previamente, este derecho de adaptación *no puede ser transferido a otros juntamente con el programa* licenciado o vendido sin la expresa autorización del titular de los derechos de autor de la obra original (…) la comparación de esta práctica -modificar un programa- con tomar apuntes en un libro es apropiada: apuntar un libro es ciertamente una obra derivada, pero salvo que el apuntador intente copiar o venderla, el titular de la obra original difícilmente estará preocupado".

7.2.3. Agotamiento del derecho de distribución.

El artículo 18 inciso final prescribe que: "con todo, la primera venta u otra transferencia de propiedad en Chile o el extranjero, agota el derecho de distribución nacional e internacional con respecto del original o ejemplar transferido". Esta disposición viene a dar sentido a la oración presente en la definición de derecho de distribución que hacía el mismo artículo 18 letra e, en la parte que dice que las copias de las obras protegidas que abarca dicho derecho son aquellas que "*no hayan sido objeto de una venta u otra transferencia de propiedad autorizada*".

Conocida en doctrina comparada como la *first sale doctrine* o *exhaustion rule* y, a efectos de esta parte del ensayo, como "agotamiento", y pese a tener distintos contornos y aplicación, se trata de una limitación al derecho de distribución y que se funda en el legítimo acceso por parte del público de las obras protegidas, y que se halla implícito en el espíritu de los sistemas de derecho de autor. En el caso preciso del agotamiento del derecho de distribución, este acceso a la obra que le correspondería al público se da mediante la generación espontánea de un mercado secundario de las copias de la obra protegida, el cual se estima que debe escapar del control del autor. Plana (2016) traza una muy interesante relación del agotamiento con respecto al sistema civil y sus principios, ya que sostiene que: "la teoría del agotamiento del derecho es la respuesta del derecho de autor frente al principio general de la libre circulación de bienes"[198]. Nótese que esta idea parece sino reafirmar que el agotamiento solo

[197] Ibid.
[198] Plana, J. (2016). *El agotamiento del derecho de distribución y su aplicación en un entorno digital. Revista Chilena de Derecho y Tecnología (Vol. 5, No. 2)*, p. 20.

se aplica con respecto a las cosas susceptibles de propiedad civil (los bienes a que apunta el art. 565 del Código Civil), vale decir, los ámbitos tangibles o soportes materiales de la obra.

El contorno general del nuestra regla de agotamiento se basa en que esta tiene extensión internacional ("en Chile o el extranjero"; "derecho de distribución nacional o internacional"). Tal como comprende Plana (2016), cuando un titular de derechos de autor "una vez introduce en el mercado un producto protegido, no puede oponerse a ventas futuras de estos productos, independiente del país en que haya sido hecha esta introducción"[199]. Además, debe advertirse que, de la parte final del art. 18 letra e, el agotamiento solo produce su efecto limitativo si el producto se obtiene de un *distribuidor autorizado.*

Determinado el contorno general de aplicación y siendo consistentes con el planteamiento del presente ensayo, es posible detectar que el estado y alcance efectivo del agotamiento chileno es reducido o anulado en todo contexto donde la información es objeto de actos jurídicos, situación aplicable a la industria de software, ya que sobre la información no es posible hablar "ventas" o "transferencias de propiedad o posesión". Esta conclusión se extiende, con razón, a toda clase de contenido cuyo formato es puramente digital por ser, en efecto, información y nada más (Plana, 2016)[200]. Piénsese que un archivo computacional, sea que contenga una imagen, un video o un programa, este es, a efectos del sistema computacional en donde está almacenado, información pura y dura (en realidad, pura y "blanda", como en *software*). Una imagen digital, en su esencia es un conjunto compacto de bits que se organiza de una manera particular (*bitmap*), lo cual reconduce a un conjunto de 1s y 0s que puede duplicarse en el contexto digital *ad infinitum* y sin costo alguno. No podemos negar el valor económico y social intrínseco de la información, pero tampoco podemos tratarla tal como tratamos a una cosa física, y esto incluye el tratamiento jurídico.

Otro aspecto que reduce su aplicación apunta a la necesidad real de la regla. ¿Es posible distinguir entre el programa original de sus copias o distinguir entre copias digitales "usadas" de las que no lo están? Por el estado actual de la tecnología es que un mercado secundario termina por ser redundante de aplicar en la información, máxime si varias expresiones de la comercialización de programas tienen una base gratuita. "Reventas de información" es una cuestión totalmente artificiosa[201], que reconduce a

[199] Ibid. pp. 21-22.
[200] Ibid. pp. 11-12; 17-20; 21-29; 35-38.
[201] El "vendedor" debería: copiar el programa, divulgar (o si se prefiere "entregar") la copia al "comprador", y luego borrar la matriz.

difundir el exacto bloque de información implicado, pero a otro sujeto. Debe notarse que estas conclusiones se aplican también respecto del software si contenido en hardware contenedor aparte y que si *parece* tener existencia física. Los desarrolladores pueden perfectamente sujetar la adquisición del ámbito tangible a un acto X ("venta") y la información contenida en dicho ámbito a un acto Y ("licencia"), por la sencilla razón de que son ámbitos distintos y no intercambiables, ni aun jurídicamente, tal como se expresó en el Capítulo 4. A mi juicio esta conclusión, que puede parecer artificiosa, obedece a que la industria de software es una industria movida por la información y servicios relacionados a esta y no se basa realmente en la demanda de CD-ROMs (solo por dar un ejemplo), por ende, el desarrollador tiene interés en regular con especial celo el aspecto más importante del complejo tráfico comercial y jurídico en la industria de software (derechos de propiedad intelectual, ámbitos tangibles y la información en sí). De hecho, han sido los propios usuarios de programas y contenido digital, los que ha pujado para que las transacciones de información limiten el ámbito tangible al mínimo imprescindible (distribución digital), de modo que la distinción entre software si contenido en hardware del que no lo es, resulta estéril y terminará yendo en contra de los intereses de desarrolladores y de los usuarios.

Los autores chilenos podrían incluso comprender legalmente a la distribución digital como un acto propio de comunicación pública, habida cuenta que no existe distribución de "copias de programa", definidas por la LPICh necesariamente como ámbitos físicos, y sortear de otra manera la aplicación del agotamiento, aunque estimo que la inutilidad o improcedencia del agotamiento en las obras en formato digital debe fundarse jurídicamente en base a intereses legítimos y no simplemente para fomentar el uso de resquicios legales para cualquier fin. Luego está el creciente fenómeno del SaaS, paradigma de comercialización relacionado más a servicios o funciones concretas de cómputo que a la dotación de información en sí, y que se alza como un fenómeno que quitará relevancia al concepto legal de distribución, en una forma lenta pero segura. Es también usual en la práctica comercial del licenciamiento de software, el hecho de sujetar a la información del programa a un acto que simplemente no pueda subsumirse como análogo a la situación de propiedad o posesión (ej. el acto se titula expresamente como "licencia", amplias restricciones en la utilización del programa, cláusula *licensed-not-sold*).

Un aspecto que tiene relación con lo último refiere al alcance e interés jurídico que tienen los desarrolladores de software asiduos a las formas restrictivas de intercambiar información (software propietario). Es posible considerar que estas formas restrictivas precarizan la situación del usuarios de programas y son *la causa* real del reducido alcance del agotamiento. Por ejemplo, la LPICh permite la ingeniería inversa sobre programas que se detenten legítimamente, pero no parece proscribir, al menos de manera inequívoca, que un desarrollador pueda prohibir actividades de ingeniería inversa, válidas a la luz del derecho de autor, pero como una obligación de no hacer, vale decir, disponerla como cláusula contractual. Más penosos aún, han sido las acciones judiciales que han pretendido o buscado pretender que un usuario *infringe derechos de autor* con ocasión de meramente ejecutar un programa que ha obtenido legítimamente.

La información de software (y todas las industrias basadas en contenido digital) tiene un potencial de difusión *de facto* enorme en la actualidad, diríamos que existe una oferta virtualmente infinita. En ese sentido, los sistemas de derecho de autor, aplicados a esta particular esfera económica, parecen funcionar más como un mecanismo que dispone suficiente escasez artificial a efectos de que no se desincentive la entrada a un mercado con relevancia manifiesta, en contraste a alzarse como un sistema que vela directamente por el acceso general de la información, ya que esto es un efecto del avance tecnológico. El fenómeno del FOSS, por ejemplo, tiene como una de sus directrices principales, la libre distribución del programa, entre otras. En estas fórmulas no restrictivas de intercambiar información, no se hace necesario, a efectos prácticos, entrar a analizar la procedencia del agotamiento. Es altamente improbable que un desarrollador de FOSS y el movimiento FOSS en general, que puede verse como un agente de suma importancia a efectos de perseguir el preciado bien jurídico del acceso libre a la información de software, esté preocupado del agotamiento de sus derechos de distribución.

Generar una teoría de cosas digitales equiparada a las cosas tangibles susceptibles de actos como ventas y arriendos, todo para estructurar un argumento que, por ejemplo, permita hacer aplicable el agotamiento del derecho de distribución, como la utilizada en el caso *Used Soft GmbH v. Oracle International Corp.* (2012, CJEU), no apunta directamente al problema del libre acceso a la información, también puede ser utilizado como excusa para generar modelos de negocios artificiosos, y genera demasiados problemas conceptuales de relevancia jurídica (ej. equiparar la información a una cosa, aplicar situaciones jurídicas extrañas a los programas y no utilizadas, sea por las fórmulas restrictivas o

más abiertas de intercambiar información). Equipar la información a una cosa, implica, implícitamente, tratarla como un bien finito o escaso en oposición a su naturaleza real presente en la actualidad. Es paradójico que se busque proteger o reafirmar la libre difusión de la información, un principio que se basa en la naturaleza de la información como un bien humano infinito, a través de una instancia que, jurídicamente, la dispondrá como bien tangible esencialmente escaso. Contrástese esta situación con derechamente fomentar mayores instancias de FOSS y garantizar la seguridad jurídica de los distintos derechos de autor, porque, mal que mal, tanto el software propietario y restrictivo como el FOSS, se basan y requieren de un sistema de derecho de autor robusto.

Se concluye así, por el tenor literal de la LPICh, la situación y escenario efectivos en que se producen los intercambios de información, y la falta de un interés jurídico real, que el agotamiento del derecho de distribución en los programas como obras protegibles por nuestro sistema, no debiera recibir aplicación, salvo con respecto a los contenedores físicos, que así se hacen aptos para reventas y donde es posible aplicar el concepto de propiedad civil y las otras situaciones jurídicas relacionadas, como poseedor y tenedor, en el sentido legal que contempla el art. 582 del Código Civil de manera expresa.

7.3. Derecho de adaptación, transformación o modificación.

El artículo 5 letra w define *transformación* como "todo acto de modificación de la obra, comprendida su traducción, adaptación y cualquier otra variación en su forma de la que se derive una obra diferente". Por la letra i del mismo artículo se define a su turno el concepto de *obra derivada* como "aquella que resulte de la adaptación, traducción u otra transformación de una obra originaria, siempre que constituya una creación autónoma". Por su parte, el artículo 18 letra c sostiene que "solo el titular del derecho de autor o quienes estuvieren expresamente autorizados por el, tendrán el derecho de (…) adaptarla a otro género, o utilizarla en cualquier otra forma que entrañe una variación, adaptación o transformación de la obra originaria, incluida la traducción".

En general, la delimitación del derecho de adaptación equivale a la determinación del alcance del concepto legal de obra derivada. En la LPICh este concepto es suficientemente adaptable al entorno digital, pero carece de detalles que hubiesen venido bien, como una enunciación de ejemplos más prácticos. Se puede agregar que, como todo derecho de autor patrimonial, el derecho de adaptación se alza como un medio legal que permite a todo autor obtener un margen adicional de ganancias económicas legítimas, y que trasciende la simple copia y difusión de la obra original.

7.3.1. Traducción del código.

Como nuestra ley protege los programas, indistintamente, "como programa fuente o programa objeto", es dable concluir que el código fuente y el código objeto son para la ley dos caras de una misma moneda, viniendo a ser esta moneda, la obra original. Por esta razón, compilar el código fuente a código objeto, no es realmente un acto constitutivo de adaptación, ya que el código resultante se encuentra protegido como otro aspecto de la obra original, constituyendo entonces un acto de reproducción. Lo mismo puede decirse de la operación inversa o descompilación del código objeto a código fuente entendible, aunque este proceso, que comúnmente no produce un resultado completamente satisfactorio[202], no es esencial para la ejecución de un programa y está más relacionado a actividades de ingeniería inversa.

Un autor puede estar interesado en traducir el código fuente a un lenguaje de idéntica abstracción o a lenguaje ensamblador por ejemplo. Esta clase de actividades se realiza usualmente con la ayuda de programas compiladores con denominación especial (*transpiladores)*. La traducción a distintos lenguajes de programación sino se considera como tal, esto es, un acto de traducción en la definición de la RAE ("pasar [algo] de una lengua o forma de expresión a otra") debe considerarse como una forma de adaptación, esta vez en el sentido de transformación o modificación de la obra original, situación que, en todo caso, se comprende en el derecho de adaptación a la luz de la LPICh.

También puede argumentarse que en toda clase de traducción, incluso en el caso de la transpilación (traducción dispuesta por un programa), estamos siempre y en todo caso, en presencia de un acto de reproducción, ya que una obra derivada debe ser "autónoma" con respecto de la obra original (debe tener su propia expresión original), y una traducción dispuesta casi totalmente por un programa autónomo y de manera totalmente mecanizada (me refiero a un compilador), difícilmente puede equipararse a las traducciones que hace un humano respecto de obras más tradicionales como un libro y, en todo caso, valiéndose totalmente de su esfuerzo intelectual y no de las funciones de un programa que hace todo el trabajo por él. Esta es la idea que probablemente subyace en la comprensión legal de nuestra LPICh de que el código objeto es el código fuente, en el sentido de que el compilador torna trivial, creativamente hablando, los actos de traducción de esta particular clase de obras. En ese sentido, si la traducción del código toma la forma de *portar* el código escrito en un lenguaje de programación

[202] Uno de los TPMs más utilizados consiste precisamente en la ofuscación del código, todo para que el interesado en descompilar el código objeto disponible obtenga un resultado casi igual de ininteligible a este último.

"X" a un lenguaje de programación "Y", y tal acto se realiza por el programador en forma esmerada y no mecanizada por un compilador (es común el concepto de *porting)* es que estimo que, la traducción del código puede dar pie a una auténtica obra derivada al tenor de la LPICh.

7.3.2. Modificación del código.

La modificación o edición del código cobra una especial relevancia en el desarrollo de cualquier programa y se relaciona a la *fase de mantenimiento*. El mantenimiento de software alude a la modificación del código de un programa preexistente, esto es y generalmente, un *programa terminado y ya sacado al mercado*, y que obedece a razones tales como: corregir errores, optimizar el diseño, estructura o rendimiento del programa (ej. mayor eficiencia en el uso de recursos), agregar o sustraer funcionalidades, y adaptar el programa a nuevos requerimientos o estándares de hardware y/o software (ej. sistema operativo). La relevancia y extensión del mantenimiento para la industria del software conlleva, en la práctica, a la salida al mercado de distintas versiones de un programa bajo distintas denominaciones según criterio del propio desarrollador: "parches", actualizaciones, correcciones.

Los actos de modificación del código del programa, no se identifican totalmente con el mantenimiento, esta siendo solo la cara más visible de los actos modificatorios, pero ¿Es toda modificación una obra derivada? Según la Circular 61 de la *Copyright Office* de Estados Unidos[203], cada modificación del programa, dada por versiones autónomas, es en principio, valga la redundancia, una obra autónoma, tanto del programa primitivo así modificado, como de futuras versiones o modificaciones. Además de presentar un problema práctico manifiesto, esto implica que la protección que pudiese recaer sobre un programa en una versión determinada no se extiende necesariamente a otras versiones, sean pasadas o futuras. Sin perjuicio de esta importante afirmación, que bien puede servir para propósitos prácticos, la interpretación de la *Copyright Office* asume sin más que toda modificación del código, acepción fáctica, *es* una obra derivada, concepto jurídico.

Siguiendo a Ravicher (2002)[205], quien utiliza la Sección 102(b) del 17 U.S. Code que define *obra derivada* en dicho sistema, diremos, desde ya, que solo un cierto conjunto de modificaciones se halla

[203] Vid. supra nota [142].
[205] Ravicher, D. (2002). *Software Derivative Work: A Jurisdiction Dependent Determination*. Obtenido de *Linux.com:* https://www.linux.com/news/software-derivative-work-jurisdiction-dependent-determination/

comprendido en el concepto de obra derivada en un programa. Tal vez el aspecto clave de esta problemática está en el concepto que emplea dicha disposición que prescribe que "una obra derivada es una obra *basada* en una o más obra preexistentes" y en la naturaleza autónoma, como obra, de la obra derivada con respecto a la originaria, situación esta última que consagra nuestra propia LPICh. Siguiendo la misma Circular 61, esta misma concede que se excluirían las versiones que entrañen cambios o revisiones *menores*. Dicho documento también agrega los cambios o revisiones determinados por "la funcionalidad del hardware". Lo relevante aquí no es la determinación de conceptos vagos como "cambio sustancial" o "cambio menor", ya que en realidad apuntaría a que una obra derivada, esta vez según la propia LPICh, debe constituir "una creación autónoma", vale decir, una obra susceptible de protección propia.

Las versiones del programa que no entrañen un sustrato protegible propio, esto es, que no contengan "expresión original", no se subsumen bajo el concepto de obra derivada y formarían una reiteración o "copia" del programa original. Bajo esta interpretación, se torna comprensible el hecho de que los cambios o revisiones fundados para adaptar al programa a la "funcionalidad del hardware" también se excluyan del ámbito de obra derivada, ya que la funcionalidad del hardware, así como otras exigencias de interoperabilidad (ej. adaptaciones en función del sistema operativo), son factores externos, aspectos que se filtran y no pasan a conformar parte del sustrato protegible de un programa, sea como obra originaria o para constituir una nueva derivada. Esta idea toma claramente la jurisprudencia del caso *Altai* y el método AFC en general. Relacionada ahora a nuestro sistema, esta interpretación es consistente con las limitantes que dispone el artículo 71 Ñ letra a y c, las cuales se verán más adelante y a propósito de las limitaciones al derecho de adaptación.

De esta manera, en este ensayo se aboga, tal como parece hacerlo agudamente Ravicher (2002)[206], por la aplicación del método *AFC* o de un necesario filtro de elementos no protegibles, para determinar si existe expresión original, a efectos de diferenciar modificaciones de programas que se homologuen al concepto de obra derivada de las que no. En esencia, el método *AFC* busca determinar la presencia o no de expresión original en un programa que es el objeto real del derecho de autor. Con todo, a mi juicio, el diseño modular, aspecto que se hace necesario admitir a efectos del método *AFC o* de cualquier intento jurisprudencial de comprender a un programa como una obra efectivamente literaria, presenta

[206] Ibid.

problemas propios en casos que un programa final complejo, vale decir, una programa vinculado a un catálogo de varias funciones y no solamente una (como el grueso de las *aplicaciones* de relevancia comercial).

Es una interpretación consistente con la práctica de la programación el considerar que un *programa final* es, en variable medida, una combinación de programas más pequeños o subprogramas interconectados (diseño modular). Cada uno de estos subprogramas puede constituir una propia combinación y/o agregación, y así sucesivamente. Si no hay más autores implicados (ej. el programa final está dado por componentes que son todos de autoría de una misma persona) no hay mayor análisis en esta parte. Sin embargo, cuando concurren dos o más autores distintos, como el caso de modificaciones realizadas en una obra primigenia por un extraño, el análisis se debe afinar.

Tratando de aterrizar esta circunstancia práctica al lenguaje de nuestro sistema de derecho de autor, además del concepto de obra derivada, en los casos de modificación de programas, hay que tener a mano los siguientes conceptos presentes en nuestra LPICh: a) compilación de datos o simplemente "compilaciones", definido en el art. 3 N° 17; b) obra en colaboración, definido en el art. 5 letra b; y c) obra colectiva, definido en la letra c de este último. Sea un programa completo o un ámbito delimitado de este, es que podemos hallarnos en cualquiera de las situaciones mencionadas, o incluso todas, ya que el análisis de abstracción comprende todos los elementos, uno por uno, y con intención de separarlos de cada uno. En ese sentido, por ejemplo, cuando dos elementos abstraídos que son de autoría de personas naturales distintas "no pueden ser separados", la combinación constituye una *obra en colaboración,* siempre que exista intención de ambos de que así sea. Pero si dichos elementos abstraídos provenientes de autores distintos si pueden ser separados, estaríamos en presencia, en general, de una compilación, y reuniendo los convenientes requisitos del art. 5 letra c, de una obra colectiva en especial[207].

La primera conclusión en torno al alcance de las modificaciones del programa respecto al derecho de adaptación sostiene que solo las modificaciones que contengan expresión original son obras derivadas en su concepto legal propio. Continuando la argumentación y suponiendo una modificación a un

[207]Esta interpretación es discutible. En la Sección 107 del 17 U.S. Code, se halla implícita la relación género-especie entre una compilación y una obra colectiva. En cualquier caso, ambas difieren notablemente de las obras en colaboración, caracterizadas por la intención de que exista coautoría sobre la totalidad de la obra.

programa que contenga sustrato protegible, la naturaleza técnica de la modificación no parece ser necesariamente un aspecto decidor para estar en presencia o no de una obra derivada. Por ejemplo, usualmente cuando hablamos de una "mejora" a un programa, nos referimos a la agregación de funcionalidades que no estaban presentes. Esto último puede implicar, ciertamente, la agregación de un componente nuevo, vale decir, la *agregación de otro programa* que en su momento no existía en el programa original. Sin embargo, dicha mejora también puede implicar una cosa distinta: una *optimización* o *potenciamiento, de un componente ya existente*. En este último caso es patente la situación de que dicha modificación, existiendo expresión original protegible, entraña una obra derivada (sin perjuicio de la aplicación de las normas relativas al problema de pluralidad de autores): diríamos que esta modificación está efectivamente *basada* en este componente. Pero en el primer caso (simple agregación) la situación es un poco más compleja. Si se mantiene la intención de *separación* del componente agregado con respecto al programa final, diríamos que el acto de modificación entraña, en principio, una simple "contribución" al programa final en su versión primitiva que, sin perjuicio de que no alterará los derechos del autor de esta última, difícilmente puede considerarse que constituye una auténtica obra derivada, ya que estamos en presencia de programas independientes que se agregan, vale decir, una compilación o una obra colectiva, según corresponda.

Existe clara intención de separabilidad, a mi juicio, por ejemplo, en el caso que la modificación consista en código convenientemente reutilizable por distintos programas ya que estas, en su propio concepto, no buscan agregarse a un programa particular o específico, sino más bien, están *especialmente diseñadas* para utilizarse por otros programas no específicamente determinados. Esta idea apunta principalmente al concepto técnico en programación de librerías o *libraries*. El programa que utiliza la librería difícilmente está *basándose* en la librería utilizada, en el sentido que se trata de "obras" o "productos" sin mayor relación. Esta es la única manera consistente de interpretar el concepto de obra colectiva y obra derivada, como conceptos propios de derecho de autor. Distinto situación sería que el programador desarrolle, en clave de Rosen & Einschlag (2004) [208], una versión "mejorada" o "potenciada", esta vez, de la librería misma, ya que solo en este caso, podríamos decir que el programador está basándose en la librería, en tanto obra. Nótese entonces que lo decidor es la relación de los programas implicados (ej. librería y el programa que la utiliza) *como obras o productos*, sin cobrar

[208] Rosen, L., & Einschlag, M. B. (2004). *Derivative Works*. Obtenido de *Rosenlaw & Einschlag:* https://www.rosenlaw.com/lj19.htm

tanta relevancia la forma técnica o los aspectos funcionales que rigen el cómo dicha relación se logra (ej. enlazamiento estático o *static linking*). Confluye en esta línea argumentativa también lo dispuesto en el art. 2 del Tratado OMPI de 1996 ya analizado (funciones y métodos no son protegibles).

En caso de no se mantenga la separación de los programas implicados en tanto obras, aún si lo que el programador hace es efectivamente *agregar* un componente y no potenciar uno existente, se produce aquí una auténtica mejora, esta vez no de un componente específico del programa final, sino del programa final en sí o su estructura, ámbito también protegible en un nivel mayor de abstracción, siendo dicha versión mejorada una posible *obra derivada* de este último. Otro ejemplo de obra derivada, según este análisis y que grafica como la forma o la naturaleza de la modificación no es tan decidora a efectos de determinar el alcance conceptual, son las versiones de programas libres de medidas de protección tecnológica (TPMs), circunstancia que puede comprenderse, muchas veces, como una *sustracción de funciones*.

La interpretación propuesta de la modificación del código como obra derivada, es consistente con el concepto de "expresión original" y también puede sortear un posible análisis de tipo económico, sea de los conceptos legales de obra derivada o de los intereses jurídicos presentes en casos concretos, todo de manera satisfactoria[209]. Piénsese en como una versión libre de TPMs de un programa, de no autorizarse, es a todas luces lesiva de los intereses patrimoniales del desarrollador. Ahora piénsese como la mera utilización de una librería accesoria (ej. funciones matemáticas como raíz cuadrada, logaritmo, etcétera) en un programa particular (ej. un procesador de texto), de considerarse que al unirse por enlazamiento u otra vía que modifique el código, pasa a constituir auténtica obtención de una obra derivada, terminaría por obstaculizar el desarrollo de programas y, de paso, a la innovación en la propia industria de software. En estos casos, los desarrolladores se verían en la necesidad de "reinventar la rueda" continuamente, frustrando la utilización de programas destinados a servir a otros, tiempo que hubiese sido mejor destinar al desarrollo del programa realmente buscado y las satisfacción

[209]En el caso *Midway* (1983), mencionado a propósito de la protección de *displays* de un programa, el tribunal sostuvo que: "Acelerar el curso de un videojuego lo torna más desafiante y emocionante e incrementa las ganancias del licenciatario por ejemplar (…) No resulta obvio de esta definición -definición de obra derivada en el 17 U.S. Code- que la aceleración del curso de un videojuego es una obra derivada. Una versión acelerada de un fonógrafo probablemente no lo es (…) pero esto es porque el *valor adicional* explotable por el titular es demasiado trivial (…) Una versión acelerada de un videojuego es un producto sustancialmente distinto al juego original. Como se ha notado, es más emocionante de jugar y requiere esfuerzo creativo para producirse".

de necesidades. Esto terminaría por obstaculizar el desarrollo de programas, horadar el principio de interoperabilidad sin real necesidad y, de paso, a la innovación en la propia industria de software.

Consistentemente con lo anterior, compartiéndose la opinión de Rosen & Einschlag (2004)[210], a efectos de colaborar en uno de los temas sorpresivamente más candentes relacionada al alcance del concepto de obra derivada aplicada a los programas, y que trata sobre la aptitud del enlazamiento y sus distintas variantes técnicas para formar una obra derivada (Rosen, 2004)[211], estimo que, es irrelevante la discusión en torno a si el enlazamiento en sí, o si alguna de sus formas específicas (estático en tiempo de compilación o dinámico en tiempo de carga o ejecución), entraña algún aspecto especial o digno de elaborar en detalle con respecto al tema de las obras derivadas en un programa. En ese mismo sentido, otra situación que tampoco cobra relevancia, al menos respecto al tema en cuestión, está en la distribución de programas sin mayor relación en "paquetes", vale decir, contenidos en un mismo medio físico o en una misma carpeta digital.

Debe reiterarse que aún en las obras colectivas o compilaciones, se debe contar con la aquiescencia de los otros autores implicados de dichas obras así agregadas, en base a los derechos exclusivos que les competen, algo que destaca el art. 3 N° 17 de la LPICh referido. ¿Por qué molestarse en distinguir las distintas situaciones posibles si tanto la mera copia como las creación de obras derivadas son derechos exclusivos del autor? La respuesta, al menos respecto de los programas, se halla en el fenómeno de las licencias recíprocas en general, y en las licencias libres con *copyleft* en especial y que se habían mencionado en este ensayo.

Las licencias libres con *copyleft* son licencias recíprocas que imponen que *las obras derivadas* de la obra original así licenciada, deben sujetarse a la misma licencia y no a otra. Las licencias libres con *copyleft,* como la ampliamente utilizada *GPL v. 2*, autorizan de manera generalizada el ejercicio de los distintos derechos de autor patrimoniales implicados en la obra originaria, pero involucran, como condición expresa del otorgamiento de la licencia, la restricción antedicha, la cual tiene aplicación, generalmente, cuando se está en presencia de una obra derivada. De esta manera, solo ante la existencia de una obra derivada del programa, es que el autor de la misma, se ve en la necesidad jurídica de licenciarla bajo los

[210] Ob. cit. supra nota [208].
[211] Rosen, L. (2004). *Open Source Licensing: Software Freedom and Intellectual Property Law.* Prentice Hall. pp. 103-107; 115-125.

términos libres que regían a la original, ya que si no lo hace, su conducta estaría fuera del ámbito de la licencia de adaptación conferida, infringiendo el derecho del autor original[212], o bien, estaría vulnerando una obligación contractual, todo dependiendo de la naturaleza jurídica que le asignemos a la cláusula de *copyleft*. En ese sentido, los contornos jurídicos precisos del copyleft aún esperan un precedente, pero en cualquiera de sus potenciales efectos, lo relevante radica en la existencia o no de una obra derivada, aunque en ciertas cláusulas de *copyleft* se utiliza el término, más general, de *modificación del programa*, circunstancia que nos obliga, según la conclusión de que no toda modificación es una obra derivada, a analizar de manera más fina los posibles efectos jurídicos.

En ese sentido, y aunando todas las conclusiones anteriores, me parece reductiva la interpretación que se hace por parte de la *Free Software Foundation* (FSF) al sostener que:

"¿Qué determina la combinación de dos partes para formar un programa? (…) a nuestro juicio, un criterio adecuado depende del *mecanismo* de comunicación (*exec*, tuberías, *rpc*, invocación de funciones dentro de un espacio de direcciones compartido, etc.) y de la semántica de la comunicación (qué tipo de información se intercambia)"[213].

El primer elemento aludido hace demasiado énfasis en cuestiones técnicas algo que, bien puede comprenderse como modificación, pero difícilmente como obra derivada, concepto este último propio del sistema de derecho de autor, ordenamiento que no busca proteger funciones o métodos, sino expresión original. Muchos de los aspectos técnicos relacionados a la forma en que dos o más programas se interrelacionan técnicamente, son aspectos muy condicionados a las prácticas asentadas de la disciplina de programación[214], o tienen relación con la eficiencia del programa o el logro de interoperabilidad funcional en sus distintos ámbitos (esto amerita recordar los ejemplos propuestos en el caso *Altai*). Se trata entonces de aspectos muy susceptibles de exclusión según el test de filtración y que obliga, en nuestro caso, tomar en cuenta el art. 2 del Tratado OMPI de 1996, esto por no entrañar real despliegue creativo por parte del programador y tener demasiada relación con las funciones o

---

[212]Esto porque el licenciatario *solo puede crear obras derivadas en tanto las distribuya de cierta manera* (en clave de la jurisprudencia norteamericana, diríamos que es una *license condition* y no un *covenant*). Esto implica que la distribución de obras derivadas en maneras distintas está *fuera* de la licencia y dan motivo para accionar según infracción de derechos de autor. Todos esto es perfectamente discutible.

[213] *Preguntas frecuentes acerca de la versión 2 de la GPL de GNU. Free Software Foundation, Inc* (2001-2020). Disponible en línea: https://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html, consultada el 5 de marzo de 2020.

[214]En ese sentido, según la FSF, *ibídem*: "por contra, tuberías, *sockets* y argumentos en la línea de órdenes *son mecanismos de comunicación habitualmente utilizados entre dos programas independientes*".

métodos descritos en el código, lo cual nos permite concluir, con alto margen de plausibilidad, que tales "mecanismos de comunicación", en sus diversas fórmulas técnicas, no constituyen un aspecto siquiera protegible por el derecho de autor, o aportan un criterio vago para determinar la presencia de una obra derivada.

En cuanto al concepto de "semántica de la comunicación", la FSF sostiene que:

"Si la semántica de la comunicación es lo bastante íntima, teniendo lugar un *intercambio de estructuras de datos internos complejos*, eso podría también servir de base para considerar que las partes están combinadas dentro de un programa mayor".

Si una modificación da pie a considerar que un programa final se ve efectivamente modificado y no a que un programa se agrega a otro, es una situación que ya se comprendió en este ensayo. En ese sentido, la FSF comprende atinadamente que solo en el primer caso (dos o más componentes se conciben como un programa general único) corresponde hablar de una potencial obra derivada (y en el otro, una compilación). Se trata de un criterio, si bien abstracto, mucho más plausible de comprender en el alcance del concepto de obra derivada, ya que alude a la relación, fundamentalmente recíproca, y en todo caso, *de fondo* que hay entre dos o más programas. Tal vez el ejemplo más típico es la *dependencia entre programas*, esto es, una *semántica* especialmente intensa al punto de total interdependencia, circunstancia que conlleva a comprender que en realidad existe un único programa, al menos al analizarse de un nivel de abstracción general. Con todo, salvo en la completa interdependencia de programas, es imprescindible un análisis más fino para determinar si estamos realmente en presencia de un programa único fusionado. También es problemático el concepto de dependencia propuesto aquí. Un programa puede efectivamente depender totalmente de ciertos programas, pero puede que dicha dependencia no sea de respecto de un programa específico y determinado, sino, por ejemplo, de una *clase de programas.* También es siempre necesario ubicar un análisis económico, ad hoc, de los programas implicados en el caso concreto. Las aplicaciones, por ejemplo, siempre requieren de la presencia de un sistema operativo, pero no de una clase o tipo particular. Aún en casos que una aplicación solo pueda ejecutarse en una clase muy particular de sistema operativo, malamente podremos decir que dicha aplicación se fusiona con el sistema operativo como un todo, al punto que da nacimiento a un programa único y que los *programadores de aplicaciones crean obras derivadas de sistemas operativos.* En realidad, este concepto de la "semántica"

termina por ser otra ilusión conceptual dada por los alcances técnicos de los programas (ej. funciones no protegibles), ya que para estar en presencia de obras derivadas se requiere, como base, alguna clase de modificación creativa, luego, aún en programas que presenten interdependencia muy intensa (como las aplicaciones con el sistema operativo), puede que no exista modificación alguna de las obras o que la modificación obedezca al funcionamiento normal de uno u otro programa.

Aún si dos o más programas de distintos autores constituyen efectivamente un programa único más grande (ej. programas A, B y C, todos de distintos autores, se conciben como un programa final o producto final "D"), para estar realmente implicado el derecho de adaptación (ej. D es un derivado de A), los programas así fusionados deben contener la relación propia de obra derivada (modificación creativa), situación que puede o no darse, según los criterios propuestos. Esto contrasta con la situación de la simple agregación de programas (en nuestro ejemplo el programa final sería aquí, un compilado: "A y B y C", y no un programa final distintivo). Aún una relación unilateral o recíproca muy íntima entre programas puede no necesariamente entrañar la creación de una obra derivada, aunque permite fundar que ciertamente estamos en presencia de un único programa. Puede ocurrir que, en nuestro ejemplo y por algún motivo, el programa D, como obra susceptible de derecho de autor, sea un *producto totalmente distinto* de "A" o "B" o "C", o cualquier combinación de estas, quizás por la alta sinergia y procesamiento logrados, circunstancia que entrañará que "D" sea, en realidad, una obra en colaboración, siempre y cuando se cumplan sus elementos legales propios y a que aluden el art. 5 letra b de la LPICh. Si no se cumplen los requisitos antedichos, estaríamos en presencia de una clase de infracción que no tiene como objeto al derecho de adaptación, ya que será relativa a los derechos de reproducción y distribución (en nuestro ejemplo, el infractor sería el autor que, combinando el o los componentes de que si es autor, con el o los componentes de los cuales no es autor, formó la combinación D).

7.3.3. Limitantes del art. 71 Ñ en el derecho de adaptación.

El art. 71 Ñ letra a, ya visto a propósito del derecho de reproducción, también se aplica con respecto al derecho de adaptación, aunque el tratamiento de la distribución de las copias obtenidas bajo dicho artículo es diferente.

La letra b del mismo artículo permite "las actividades de ingeniería inversa sobre una copia obtenida legalmente (…) que se realicen con el único propósito de lograr la compatibilidad operativa entre programas computacionales".

Por último y en mención aparte, la letra c) dispone que:

"Las actividades que se realicen sobre una copia obtenida legalmente (…) con el único propósito de probar, investigar o corregir su funcionamiento o la seguridad de este u otros programas, de la red o del computador sobre el que se aplica".

En cierta medida, el contenido de la letra c) es redundante respecto del contenido presente en las letras a y b. Una actividad realizada con el "único propósito" de investigar el funcionamiento de un programa es un concepto bastante certero de ingeniería inversa (letra b). Una actividad que se realiza con el "único propósito" de *corregir* el funcionamiento de un programa puede identificarse, muy fácilmente, con una adaptación encaminada al uso esencial del programa (letra a). Pese a esta redundancia, las letras a y c difieren de los conceptos "tenedor" y "copia obtenida legalmente" circunstancia que, por lo expuesto en este trabajo, si es relevante.

La razón de por qué estas tres disposiciones guardan relación puede resumirse por la CONTU (1978)[215] en la siguiente frase:

"*Producto de la falta completa de estandarización respecto de lenguajes de programación y dispositivos de hardware en la industria de computación*, aquel que adquiere legítimamente una copia de un programa frecuentemente no puede utilizarlo sin adaptarlo en el preciso ámbito que le permitirá utilizarlo en su computador".

La falta de estandarización a que alude la CONTU es una temática propia del principio de interoperabilidad, el cual es relevante también con respecto a los distintos ámbitos de software (ej. compatibilidad de aplicaciones con el sistema operativo). En su reporte, la CONTU no alude únicamente a las adaptaciones que permitan la simple ejecución del programa (ej. un programa con errores sustanciales que no permiten que se ejecute seguramente en un dispositivo determinado), ya que el concepto de "uso esencial" (simple ejecución del programa), aplicado a las adaptaciones, parece excluir

[215] Ob. cit. supra nota [196] pp. 32.

modificaciones que permitan, por ejemplo, ampliar las funciones del programa, ya que estas no son imprescindibles *para ejecutarlo*. Sucede que la CONTU sostenía de manera virutalmente explícita, que actividades como, por ejemplo, "mejorar" un programa mediante la inclusión de funciones no existentes, constituía un acto de adaptación amparado por la limitante del uso esencial (la *essential step defense* del 17. U.S. Code) y no otra[216]. Esta posible extensión generalizada de la letra a, sobre cualquier clase de adaptación que pudiera concebir el usuario, observando eso si las limitaciones de distribución de las adaptaciones así obtenidas, termina por hacer redundantes las otras disposiciones del art. 71 Ñ, al menos aquel contenido relacionado a la posible libre modificación del código por parte del usuario.

De estimarse conducente las ideas presentadas del reporte de la CONTU, el art. 71 Ñ letra a, con respecto al derecho de adaptación, consagra la libre modificación del código del programa para uso personal. Este "uso" es la ejecución del programa, vale decir, el usuario puede modificar el programa a efectos de cualquier necesidad de cómputo que tenga y no alcanza los "usos" de derecho de autor (ej. adaptar el programa, luego distribuir o comunicar públicamente la modificación, sea fijando un precio o no), salvo que cuente con la autorización pertinente. Como el derecho de autor consagra la exclusividad de la "ejecución pública" de la obra, nuevamente y salvo autorización expresa, la ejecución del programa debe ser necesariamente en un contexto no público al tenor de la LPICh, de ahí que se haga hincapié en que la "libre modificación" de programas es tal solo en un contexto personal o doméstico.

Hay que recordar que la *ejecución misma del programa* y los resultados obtenidos de dicha ejecución, salvo que se comprenda en un contexto público al tenor de la LPICh, en la forma que sea y para los fines que sean, es un acto excluido del ordenamiento de derecho de autor, primero, por no formar parte del catálogo propio del art. 18 de la LPICh, y segundo, por conllevar, a lo más, una reproducción (fase de carga) que se excluye expresamente del ordenamiento en la letra a del art. 71 Ñ. Los desarrolladores pueden limitar la forma y fines de la ejecución del programa (ej. "solo fines de entretenimiento"), pero dichas limitaciones, tal como aduce la CONTU (1978)[217] y la jurisprudencia norteamericana (ej. "license

[216]Versión original (CONTU, 1978: 32): "*The conversion of a program from one higher-level language to another to facilitate use would fall within this right, as would the right to add features to the program that were not present at the time of rightful acquisition*".
[217] Ob. cit. supra nota [196] pp. 32-33.

conditions" v/s "contractual covenants") generan obligaciones contractuales para el usuario o derechos personales para el desarrollador.

Con todo lo anterior, estimo que las modificaciones seguidas según el principio de interoperabilidad merecen una mención particular. La jurisprudencia en *Altai* y *Gates* ya había excluido los ámbitos destinados a lograr compatibilidad entre programas del sustrato protegible, así como las distintas prácticas de programación asentadas. El principio de interoperabilidad, más que una práctica asentada, es el espíritu de la programación, presente en cada componente o reunión de componentes de un sistema computacional. En ese sentido, en este ensayo ya se ha explicitado que las modificaciones que no impliquen el nacimiento de una auténtica obra derivada, reconducen a una copia de la obra. Por esta situación, estas modificaciones previstas para lograr interoperabilidad, no constituyen adaptación de la obra, pudiendo alzarse como una reproducción de la obra, que es alcanzada por las letras b y c del art. 71 Ñ, vale decir, también es posible de hacer de manera libre por parte del usuario.

¿Quiere esto decir que el usuario que desarrolle una versión más interoperable de un programa tiene "chipe libre" total? La respuesta es negativa, ya que dicha versión más interoperable, al no incluir expresión original, no es una obra autónoma de la obra original, identificándose con esta. Luego, solo el autor puede distribuir o comunicar públicamente la obra original, por lo que dicha versión interoperable, en los hechos, solo podrá utilizarse para fines personales o dentro de un ámbito no enmarcado de "público". Los actos de ingeniería inversa son permitidos por los sistemas de derecho de autor, porque refieren esencialmente a las funciones del programa (investigarlas), un ámbito posible de patentar. Sobre los actos excluidos que prescribe la letra c, debe decirse que se hallan así, bajo el mismo razonamiento, ya que refieren al *correcto funcionamiento* del programa. Sea que las funciones del programa sean técnicamente buenas o malas, estas no son protegidas por el derecho de autor. Así, nuestra ley implícitamente comprende el principio de interoperabilidad en su faz más pura, esto es, en su manera correcta, como un tema relacionado íntimamente a *las funciones* de los distintos componentes presentes en un sistema computacional, incluyendo los programas, y que simplemente se agrega a la ingeniería inversa y a las exigencias de seguridad del programa, esta última siendo, nuevamente, un ámbito relacionado al funcionamiento del programa.

Un aspecto que no se abordará mayormente, relacionado con este tópico de la seguridad de un programa y la adaptación, constando que nuestro sistema de derecho de autor no se ha "potenciado"

con los derechos anti-elusión de medidas tecnológicas de protección (TPMs), hallándonos en una situación de facto, es que los usuarios pueden incluso, basándose en la misma letra a (adaptación uso esencial que es amplísima según comprende implícitamente la CONTU) o la frase "corregir *su funcionamiento o la seguridad del mismo* u otros programas" de la letra c, modificar el programa a efectos de eludir las TPMs presentes en el programa, *pero siempre, y nuevamente, para uso personal*.

Como una versión libre de TPM es una efectiva obra derivada (adaptación), si el usuario intenta difundir dicha versión libre de TPMs, careciendo del permiso competente, vulnera la limitante de distribución del art. 71 Ñ letra a en la parte que limita la distribución de adaptaciones. Este usuario tampoco podría ampararse en la limitante de la letra b (ingeniería inversa a efectos de desarrollo) ya que está efectivamente comercializando un programa que atenta contra la LPICh (derecho de adaptación exclusivo del autor del programa) o, en cualquier caso, infringiendo otros derechos del autor (comunicación pública).

El problema o vacío legal propio de la falta de un sistema anti-elusión, y del que precisamente estos se hacen cargo[218], está en la comercialización o difusión de la tecnología de elusión en sí. En nuestro ejemplo y en el contexto de derecho de autor, poco parece impedir que el usuario que haya descubierto la forma de eludir una o más TPMs de un programa, difunda dicha fórmula, incluso por un precio, esto en contraste con difundir directamente la versión modificada del programa original, situación que sí tiene consecuencias jurídicas al relacionarse intensamente al derecho de autor de adaptación. Sin perjuicio de esta aparente falta de instrumentos, el desarrollador podría argüir, siendo discutible la procedencia real, que sus derechos de adaptación han sido vulnerados, pero con respecto de las TPMs mismas, ya que estas son usualmente programas. Sin embargo, es discutible que un programa destinado a eludir a otro programa se alce como "obra derivada" del mismo programa así eludido (ej. ¿puede decirse que el primero está "basado" o nace de una transformación del segundo?). Por de pronto, los desarrolladores nacionales, de necesitarlo, tendrían que recurrir a otro sistema de protección sus intereses presentes en el programa sujeto a TPM[220].

[218]Véase especialmente la Sección 1201 (a)(2) y (b)(1) de la DMCA (1998).

[220]Si hay disposiciones contractuales que prohíban la destrucción o inutilización de las TPMs por parte del usuario, puede proceder, contra el comercializador de las tecnologías de elusión, el art. 4 letra f de la Ley 20.169 de Competencia Desleal, todo sin perjuicio de las posibles infracciones a las marcas comerciales del desarrollador.

7.4. Derecho de comunicación y ejecución públicas.

Tanto la comunicación pública como la ejecución pública, y el concepto de "publicación de la obra", aluden sino a la difusión o diseminación de la obra en una manera que comprometa razonablemente el interés patrimonial del autor.

Siguiendo las disposiciones del art. 5 letras o y v y del art. 18 letras a y d, diremos de hecho que la "publicación de la obra" es una clase particular de comunicación pública. Como el art 18 letra a también dispone que "solo el titular del derecho de autor (…) tendrán el derecho de utilizar la obra en alguna de las siguientes formas: a) publicarla mediante su (…) ejecución, lectura, recitación, exhibición, y, en general, *cualquier otro medio de comunicación al público*", diremos que la ejecución pública de la obra, concepto que no se aborda por la LPICh y que reconduce a su interpretación según su lenguaje ordinario, es otra especie de comunicación pública.

En ese sentido, la definición legal de comunicación pública de nuestra ley es la más general y completa de las vistas. La art. 5 letra o ya mencionado sostiene que la comunicación pública consiste en: "todo acto, ejecutado por cualquier medio o procedimiento que sirva para *difundir los signos, palabras, los sonidos o las imágenes, actualmente conocido o que se conozca en el futuro, por el cual una pluralidad de personas, reunidas o no en un mismo lugar, pueda tener acceso a la obra sin distribución previa de ejemplares a cada una de ellas*, incluyendo la puesta a disposición de la obra al público, de forma tal que los miembros del público puedan acceder a ella desde el lugar y en el momento que cada uno de ellos elija". El contexto público, que completa el concepto de comunicación pública, está dado en la misma disposición: "una pluralidad de personas, reunidas o no en un mismo lugar".

Se trata de un concepto amplio y que se relaciona de manera íntima con el proceso de difusión que atraviesa la información de toda clase. A diferencia de la distribución y sin perjuicio del alcance del la oración "del original" presente en el art. 18 letra e de la LPICh, se trata de una difusión que no se vale de contenedores físicos que se comercializan. Si no era posible comprender a la distribución digital como un acto de distribución, por no entrañar copia tangible alguna, si es posible subsumirlo como un acto de comunicación pública. Si no es posible comprender la prestación de un software como servicio o SaaS como un acto de distribución, por no entrañar siquiera traspaso de la información en base al código del programa, si es posible subsumirlo como un acto de comunicación pública, por cuanto la art. 5 letra o dispone que "la puesta a disposición de la obra al público, de forma tal que los miembros del

público *puedan acceder a ella desde el lugar y en el momento que cada uno de ellos elija*" (cursiva propia). Si se estima que el software como servicio no entraña real difusión del programa, por cuanto nuestra LPICh limita el concepto de programa únicamente al código, y en efecto, el código no es traspasado o difundido al usuario en momento alguno en el SaaS, podemos argumentar que la obra objeto de la comunicación pública en el SaaS son los resultados audiovisuales del programa. Por esta razón, el derecho de comunicación pública también puede extenderse, en general, a los casos de transmisión o *streaming*.

En la LPICh, el concepto de comunicación pública es suficientemente amplio y adaptado a las formas en que una obra puede difundirse en una época altamente digitalizada. En el ámbito de los programas, de partida, podemos incluir la "ejecución pública" de que habla nuestra LPICh como uno de estos supuestos. Los programas son obras que pueden ciertamente ser objeto de un acto de "ejecución", en un sentido natural y obvio, la cual supondrá siempre un computador físico e implicará finalmente la obtención de resultados audiovisuales (ámbito protegido de manera separada) a partir del código o concepto legal del programa, pero observando que debe darse el contexto público.

Pero es posible ir incluso más allá, tocando por ejemplo, el fenómeno de los deportes electrónicos o *e-sports* (suponen un programa) y de cómo la realización de torneos o eventos públicos a este respecto constituyen actos que pueden fácilmente subsumirse en este amplísimo concepto de comunicación, o ejecución en su caso, con el carácter de pública[221]. En ese sentido, el derecho de comunicación pública reúne tan solo dos elementos (difusión de la obra y el contexto público), existiendo cada vez más maneras en que dichos elementos pueden configurarse. Es posible extenderlo, sin mayor argumentación, a casos más extravagantes, pero no por eso poco comunes, como formas y medios audiovisuales que muestran el programa en ejecución de alguna manera (ej. videos en plataformas o sitios web de acceso público como *YouTube*).

7.4.1. Concepto de "público" en la Ley 17.336.

Tal como se dijo, la definición de comunicación pública alude a "una pluralidad de personas, reunidas o no en mismo lugar". Sin perjuicio de esta gran extensión, debemos tener en cuenta que, según el art. 71 N de la LPICh: "no se considera comunicación ni ejecución pública de la obra, inclusive tratándose de

[221] Puede citarse la demanda ante tribunales surcoreanos que presentó Blizzard contra MBC (Munhwa Broadcasting Corporation), OGN (Ongamenet) y la KESPA (Korean eSports Association) (2010).

fonogramas, su utilización dentro del núcleo familiar, en establecimientos educacionales, de beneficencia, bibliotecas, archivos y museos, siempre que esta utilización se efectúe sin fines lucrativos. En estos casos no se requerirá autorización del autor o titular ni pago de remuneración alguna".

Se trata de una disposición clara que viene a limitar el alcance del derecho de comunicación pública (incluye ejecución) y que permite determinar los contornos del término "público" en la LPICh. Podríamos sostener que tal contexto implica que la difusión de la obra debe de ser de importancia con respecto al autor, vale decir, debe traducirse en un contexto que por su particularidades (ej. potencial de difusión alto de la obra) se relacione con algún interés patrimonial legítimo y balanceado con el interés general de acceder a la obra.

7.4.2. Limitante respecto de demostraciones a la clientela.

Según el art. 71 E inciso 2 de la LPICh:

"En el caso de los establecimientos comerciales en que se vendan equipos o programas computacionales, será libre y sin pago de remuneración la utilización de obras protegidas obtenidas lícitamente, con el exclusivo objeto de efectuar demostraciones a la clientela y en las mismas condiciones señaladas en el inciso anterior".

El inciso anterior a que refiere esta disposición dispone en la parte atingente que, estas demostraciones: "se realicen dentro del propio local o de la sección del establecimiento destinada a este objeto y en condiciones que eviten su difusión al exterior".

Nótese que la excepción solo se da en favor de ciertos establecimientos (ej. ciber cafés, tiendas de programas). Supuestamente, la disposición anterior permite toda clase actos (ej. reproducción, adaptación) relacionados a la simple demostración de la obra a una clientela, pero por la naturaleza de esta clase de actos, será respecto de la comunicación pública, particularmente la ejecución pública, el aspecto más relevante. Estas demostraciones tienen efectivamente como objeto el código del programa (es lo que el demostrador efectivamente "ejecutará"), vale decir, la obra protegida es el programa en su concepto legal, ocurriendo, sin embargo, una clara relación con otra obra susceptible de protección separada: *los displays o resultados audiovisuales*.

7.4.3. Las limitantes generales en la Ley 17.336.

El derecho de comunicación pública es el derecho quizás con mayor extensión de todos los derechos de autor. Esto se traduce en que un potencial increíblemente amplio de actos es exclusivo del autor. Nuestro subsistema de limitaciones dentro del sistema general de derecho de autor se estructura por la Ley 17.336 en la forma de disposiciones puntuales. Las posibilidades interpretativas de dichas disposiciones, es mucho menor que en un subsistema de limitaciones que se estructure en base a principios o disposiciones más generales.

Cobra así sentido enunciar en esta parte de este ensayo, las limitaciones generales de los derechos de autor, aplicadas a toda clase de obras protegidas, presentes en los arts. 71 Q, 71 K, 71 B, 71 C inciso 1°, 71 P, y 71 S.

8. Derecho de Autor Chileno en los Programas: Titularidad.

Solo el titular de los derechos de autor en un programa puede realizar los actos que enuncia el art. 18 de la LPICh y cuyo contenido se ha procurado determinar en el Capítulo 7. Según el art. 17 de la LPICh, el titular goza también de la facultad desprenderse voluntariamente de dicha titularidad mediante cesiones totales o parciales[222], así como la facultad de autorizar a terceros la realización de dichos actos, conservando en este caso la titularidad.

El problema general de la titularidad, se aborda muy genéricamente en el art. 1 de la LPICh que dispone que: "la presente ley protege los derechos que, por el solo hecho de la creación de la obra, adquieren *los autores* de obras de la inteligencia (…)". Por su parte, el art. 7 sostiene que "es titular original del derecho el autor de la obra". De esta manera, la autoría en sentido jurídico o titularidad de los derechos de autor se identificará con la autoría material, significado ordinario de autor o la acepción tercera de la definición de autor en la RAE ("persona que ha producido alguna obra científica, literaria o artística), consagrándose así también el principio de protección automática. Se halla implícita en estas disposiciones, la idea de que una obra elegible de protección es esencialmente una creación humana, de manera que, al menos de las normas aquí citadas, el titular de derechos de autor será una persona natural.

Debe siempre advertirse que en los programas cobrarán especial relevancia las disposiciones del art. 5 letras b ("obra en colaboración") y c ("obra colectiva") y el art. 3 N° 17 ("bases de datos y compilaciones"), situaciones relacionadas al diseño modular (y coautoría si cabe) y que se vio más en detalle en el Capítulo 7.3.2. Sobre el derecho de adaptación y otros alcances con respecto a la titularidad sobre obras derivadas, según lo concluido en los capítulos previos, solo el titular de dicho derecho o quien esté expresamente autorizado por aquel, puede preparar obras derivadas (ej. modificaciones creativas del código). Además, en estos mismos casos y por disposición expresa del art. 9, al publicar dicha obra derivada, el adaptador deberá procurar que figure "el nombre o seudónimo del autor original", esto en el entendido de que el derecho moral de paternidad no se extingue con la sola

[222]La "totalidad/parcialidad" aludiría, siguiendo el punto de vista de este ensayo, en el grado de amplitud de los actos implicados. También aludiría a que el titular originario puede ceder de manera específica alguno de los derechos de autor y, manteniendo la titularidad con respecto a los otros.

modificación de la obra, aún si hecha por tercero con la competente autorización. Esta misma norma también permite fundar y dejar asentada la irrenunciabilidad de los derechos extrapatrimoniales.

Por la admisión implícita de la renuncia de derechos patrimoniales que prescribe el art. 11 letra c, es que un autor puede dar lugar al software de dominio público. En contra, puede sostenerse que, por el tenor literal del art. 86 de la LPICh, la renuncia de derechos patrimoniales no produce efectos. Nuevamente en favor de la procedencia de la renuncia, puede decirse que el art. 86 citado refiere únicamente a una especie de derecho extrapatrimonial, o al menos simplemente irrenunciable, a percibir ciertas remuneraciones por la cesión de titularidad, comprensión que, de hecho, reafirman los arts. 2 y 3 de la Ley 20.243, aunque solo relativo a obras audiovisuales. De proceder la renuncia, el software de dominio público puede definirse más concretamente a efectos de nuestro sistema, del cual se dice que no admite renuncia de derechos morales por ser inalienables, como aquel programa que puede utilizarse libremente en alguna de las formas prescritas en el art. 18 de la LPICh, respetando los derechos morales del autor de la obra. De proceder la renuncia, que se entiende como acto unilateral del titular, la forma en que debe darse para que produzca efectos, es un problema aparte y tampoco regulado expresamente en la LPICh, de modo que aparece aplicable la regla general en los actos jurídicos, vale decir, no hay formalidad, aunque conviene que el interesado, manifieste su voluntad de renunciar en forma expresa y, sobre todo, clara en el sentido de renunciar y no simplemente otorgar una autorización o licencia generalizada sobre cada derecho y en todo ámbito posible, ya que nuestra LPICh no reconoce la extinción de derechos patrimoniales por la no utilización o el abandono de los mismos, ni mucho menos por la imprecisión o laxitud de las licencias, las cuales en teoría deberían cumplir con el art. 20 de la misma.

La transferencia de derechos de autor entre vivos cobra especial relevancia y dan sentido al concepto de “titular secundario” que dispone el art. 7. Sobre la naturaleza de las transferencias, conviene tener en cuenta que la disposición del art. 73 de la LPICh, dispone que, además de que la transferencia debe registrarse en el Registro de Propiedad Intelectual de cargo del Departamento de Derechos Intelectuales, fijándose incluso un plazo para materializar el registro, el acto mismo de la transferencia debe sujetarse a una especie de *solemnidad* expresa: “la transferencia deberá efectuarse por instrumento público o por instrumento privado autorizado ante notario”. Sobre estas formalidades, en ninguna parte de la LPICh se disponen las consecuencias jurídicas de la inobservancia de uno u otra, o

bien, de la naturaleza jurídica de cada una, pero es dable pensar que, por esta misma situación, se hace aplicable el art. 1682 inciso 1 del Código Civil y por ende la nulidad absoluta de la transferencia ("formalidad o requisito que las leyes prescriben para el valor de ciertos actos o contratos *en consideración a la naturaleza de ellos* y no a la calidad o estado de las personas"). Por último, y relacionado a esto, a diferencia de las cesiones, las simples autorizaciones, permisos o licencias, pueden tener, según el art. 20, "cualquier forma contractual" (ej. instrumento privado) y, en cualquier caso, no se regulan mayormente en su forma, en una disposición análoga al art. 73 de la LPICh. Sobre la posibilidad de disponer licencias exclusivas de derechos de autor, teniendo en cuenta lo sostenido en este ensayo sobre el concepto legal de programa como código, el art. 22 de la LPICh dispone que: "*las autorizaciones relativas a obras literarias* o musicales *no confieren el uso exclusivo de la obra*, manteniendo el titular la facultad de concederlo, también sin exclusividad, a terceros, salvo pacto en contrario".

Aun teniendo en cuenta que el titular de los derechos de autor será el autor material, esencialmente una persona natural, se debe tener especial consideración al art. 8 incisos 2 y 3 de la LPICh:

"Tratándose de *programas computacionales*, serán titulares del derecho de autor respectivo las personas naturales o jurídicas cuyos dependientes, en el desempeño de sus funciones laborales, los hubiesen producido, salvo estipulación escrita en contrario. Respecto de los programas computacionales producidos por encargo de un tercero, se reputarán cedidos a éste los derechos de su autor, salvo estipulación escrita en contrario".

Se trata de una declinación a la regla general prescrita por los arts. 1 y 7 de la misma LPICh solo con respecto a los *programas como código* (programa es un concepto legal), de modo que los programas desarrollados por encargo a contratistas de servicios de programación o en el marco de una relación laboral caracterizada por subordinación y dependencia, se entiende que, y salvo estipulación expresa en contrario, son cedidos al que encarga el servicio de programación o al empleador, respectivamente, y no se radican en el autor material (el programador).

Si bien la disposición es relativamente clara en el sentido de sus consecuencias, posible de relacionar a los *work-for-hire* del 17 U.S. Code y disposiciones comparadas similares, cobrando lógica que debe garantizársele al que contrata los servicios o al empleador el aprovechamiento económico de la obra y la seguridad jurídica a este respecto, siendo también relevante si se aplica esta declinación a los

derechos morales o extrapatrimoniales[224], producto de que no está totalmente claro el hecho de que la cesión tenga como antecedente el contrato entre las partes o la necesaria creación o existencia misma del programa, o bien, la necesidad de observar el requisito que prescribe el art. 73 ya visto a propósito de la transferencia de derechos de autor. Por esta situación, la determinación del alcance preciso del art. 8 inciso 2do merece algunas palabras adicionales.

En ese sentido, forzar la interpretación de que el contrato de trabajo o de servicios debe sujetarse al art. 73, al referir sobre una supuesta "cesión" para producir el efecto de que habla el art. 8 inciso 2, tornaría la norma en redundante y muy poco ajustada a los problemas prácticos (ej. las empresas de programas con un *staff* de programadores sujetos a contrato de trabajo deberían observar lo dispuesto en el art. 73). Teniendo en cuenta la ubicación de la disposición del art. 8 completo en el acápite particular de "sujetos del derecho", máxime si se tiene en cuenta que el art. 10 de la LPICh se toma la molestia de regular de manera especial la duración de los derechos *nacidos* a favor de las *personas jurídicas implicadas en los casos del art. 8 inciso 2*, es más dable concluir que, perfeccionada tanto la relación laboral como la de mero servicio, cualquier programa realizado en tales contextos, da efectivo nacimiento a los derechos de autor, al menos patrimoniales, para los allí indicados, como *titulares originarios,* y no establece efectiva *cesión* de derechos que conlleva a titularidad secundaria, ni mucho menos es un mero antecedente para que se practique alguna cesión futura eventual. En ese sentido, según la propia LPICh, una persona jurídica si puede alzarse como titular originario de derechos de autor, siempre eso sí, a propósito del código de un programa.

Existe también otra interpretación, acorde con el art. 10 inciso 1 de la LPICh que dispone la duración de los derechos de autor patrimoniales y que reza: "en el caso previsto en el inciso segundo del artículo y *siendo el empleador* una persona jurídica, la protección será de 70 años a contar desde la primera publicación". Analizando esta disposición, se puede concluir que, la titularidad originaria delimitada en el párrafo anterior, recibe efectivamente aplicación, pero solo en las relaciones laborales: programador y empleador. En el caso del arrendamiento de servicios, si bien se reputan cedidos los derechos, el titular originario es el contratista y no el arrendador (ej. duración del derecho se cuenta conforme a las reglas

[224]En el caso del inciso 3°, se utiliza el término de "cesión", situación que no armoniza con la inalienabilidad del derecho moral que prescribe el art. 16. Esto genera la convicción de que la disposición del art. 8 inciso 2 es una regla de titularidad originaria.

generales). En teoría, las partes deberían realizar los trámites del art. 73, con antecedente en el contrato de servicios, para dar pie a la titularidad secundaria en favor del arrendador. En caso de que este programador sea a su vez empleado dependiente (ej. el arrendador arrendó el servicio a *una empresa de servicios*), allí recibe aplicación el inciso 2 del art. 8, pero para luego dar aplicación al art. 10 inciso 1 a efectos de clarificar el estado de la titularidad originaria. En este último caso, la cesión en cuestión tendrá como titular originario al empleador del programador y al arrendador como titular secundario. Considero que esta es una interpretación a considerar, en tanto debe advertirse que el contrato de arriendo de servicios en el Código Civil chileno es por regla general consensual, mientras que el contrato de trabajo, si bien consensual, normalmente constará eventualmente por escrito y es más apto para de seguridad jurídica por sí solo con respecto a la titularidad originaria, todo esto conforme a los incisos 2 y 3 del art. 9 del Código del Trabajo.

9. Conclusiones.

El derecho de autor es un medio de protección jurídica de los programas con características particulares. Al menos en nuestro país, el principio de protección automática, la posibilidad de que se aplique a programas complejos o más simples y sin importar la funcionalidad, la amplitud del contenido de los derechos conferidos así como la extensión erga omnes de los mismos, y sus disposiciones especialmente adaptadas, son algunos de las bondades sustantivas particulares. Sin perjuicio de no haberse mencionado explícitamente, conviene tener presente los distintos remedios, de índole más procesal, con que cuenta nuestra LPICh en caso de infracción de derechos de autor[225], así como las acciones penales que nacen a propósito de esta misma circunstancia. Por otra parte, la naturaleza funcional de los programas, la existencia de otras posibles obras implicadas como los resultados audiovisuales de la pantalla, la falta de claridad al abordar ciertos ámbitos como la procedencia de la renuncia, la confusión típica entre hardware y software, y el contexto digital general, son algunos de los aspectos que generan desafíos interesantes o ámbitos de discusión.

En general, el fundamento particular por el cual el derecho de autor puede y debe proteger a los programas tiene que ver con que su desarrollo entraña un esfuerzo intelectual al que se le ha asignado valor espontáneo por parte de la sociedad, valor que se halla en gran medida independizado del valor asignado a la construcción y desarrollo de los computadores físicos en que los primeros se montan. Efectivamente, debe verse a la programación como una forma de escritura moderna donde existe libertad creativa, aunque bien podemos decir que tal libertad creativa es menor que la presente en obras literarias más tradicionales. Aún si la naturaleza de los programas obliga considerarlos necesariamente como información, la existencia de una industria de software independiente a la del hardware, nos obliga a concluir que la información y, más propiamente, bloques particulares y definidos de información, gozan de un valor comparable al de una cosa tangible.

El vehículo jurídico por el cual se les asigna valor y movimiento a estos verdaderos bloques de información es, tal como se ha sostenido, un acto jurídico complejo denominado *licencia de software*, el cual es casi por regla general un acto híbrido o compuesto y cuyo objeto comercial es el programa, más propiamente, la divulgación de la información en que efectivamente consiste el programa, a la par

[225] Las acciones del art. 85 B, 85 C, y 85 F, las medidas cautelares del art. 85 D, las reglas de determinación de daños especiales del art. 85 E, y la tramitación sumaria del procedimiento civil (art. 85 J).

que dispone un marco general de utilización, que tiene, nuevamente, por objeto al programa. Tal como un programa se concibe generalmente como varias partes interconectadas, una licencia de software se compone de diversos actos de naturaleza jurídica diversa (ej. cláusulas contractuales, permisos de derecho de autor, licencias de patentes), pero todas en torno a un programa determinado. En ese sentido, si hubiese un efectivo *objeto jurídico* presente en una licencia de software, diremos que trata de uno o más actos humanos, pero todos relativos al programa como información, y no al programa con ocasión de un servicio que, quizás relacionado a un programa, tiene realmente un objeto diverso. De esta manera, descargar un programa tiene por objeto la entrega de información, debiendo acompañarse una licencia, pero acceder a internet a efectos de obtener resultados y funciones de cómputo (SaaS) tiene por objeto dichos servicios y no a la información. En los servicios de programación, si bien el objeto primero es un servicio relacionado al programa, como la escrituración misma, es dable pensar que en algún momento tendría lugar la entrega de un bloque de información (el programa escrito terminado), aunque esta circunstancia parece abordarla nuestra propia LPICh en su art. 8 inciso 2.

Sobre el fenómeno del licenciamiento de software en relación al derecho de autor, puede decirse que, si el acto referido por este instrumento, en el marco de utilización del programa, se subsume como un acto propio de derecho de autor conforme a lo sostenido en el Capítulo 7, y tal acto es en todo caso permitido en favor del usuario, entonces solo allí es que estamos en presencia de una auténtica licencia de derecho de autor. De esta manera, es en la intersección entre “licencia de software” y “licencia de derecho de autor” donde se halla el fenómeno cada vez más relevante, del Free and Open-Source Software (FOSS), el cual, para ser tal, debe permitir a los usuarios modificar y distribuir el programa en tanto información, todo sin restricciones triviales. En las maneras más restrictivas de licenciamiento de software, es usual que el grueso de los derechos de autor del desarrollador se reserven celosamente, incluyendo cláusulas que incluso sobrepasen los límites del sistema. A mi juicio, en estas licencias de software, el ámbito contractual pasa a primar por sobre el verdadero tráfico jurídico de derechos de autor (generalmente en la forma de permisos y no cesiones, aunque en ciertos casos, el tráfico toma la forma de renuncia al dominio público). Será el licenciamiento de software, en general, y el FOSS bien en particular, los fenómenos económico-sociales que probarán si nuestra LPICh es un ordenamiento jurídico al servicio del “progreso de las artes y las ciencias”, pero en el ámbito preciso de los programas.

10. Blibliografía.

Atwood, J. (4 de abril de 2007). *Pick a License, Any License*. Obtenido de *Coding Horror:* https://blog.codinghorror.com/pick-a-license-any-license/

Becker, G. (2013). *On Reforming the Patent System.* Obtenido de *The Becker-Posner Blog*: https://www.becker-posner-blog.com/2013/07/on-reforming-the-patent-system-becker.html

Blind, K., & Böhm, M. (2019). *The Relationship Between Open Source Software and Standard Setting.* Publications Office of the European Union. pp. 41-45; 75-78.

C. Phillips, J. (Abril de 1992). *Sui Generis Intellectual Property Protection for Computer Software*. Obtenido de *Berkman Klein Center:* https://cyber.harvard.edu/property/protection/resources/phillips_unedited.html

Cid, J., & Merello, A. (2007). *Patentes de Invención y Software.* Santiago. pp. 59-63.

DiBona, C., Ockman, S., & Stone, M. (1999). *Open Sources: Voices from the Open Source Revolution.* O'Reilly Media. pp. 16-21; 31-37; 79-85; 96-100.

Dusollier, S. (2010). *Scoping Study on Copyright and Related Rights and the Public Domain.* Oxford: Organización Mundial de la Propiedad Intelectual (OMPI). pp. 6-7; 21-22.

Falk, J. (2018). *The Modern Epoch And The Emergence Of The Modern Calculator.* Obtenido de *Metastudies:* http://metastudies.net/pmwiki/pmwiki.php?n=Site.TheModernEpochAndTheEmergenceOfTheModern Calculator

Guadamuz, A. (2014). *Comparative Analysis of National Approaches on Voluntary Copyright Relinquishment.* Organización Mundial de la Propiedad Intelectual. pp. 7-14; 16; 17-18; 20-22.

Heath, S. A. (2005). *Contracts, Copyright, and Confusion: Revisiting the Eforceability of "Shrinkwrap" Licenses. Chicago Kent Journal of Intellectual Property*, pp. 12-20.

Hillman, R. A., & O'Rourke, M. (2010). *Principles of the Law of Software Contracts: Some Highlights. Tulane Law Review*, pp. 1521-1522; 1529-1532.

Hippel, E. v., & Krogh, G. v. (2009). *Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. Organization Science 14 (2)*, pp. 209-212.

Hojnik, J. (2017). *Technology neutral EU law: Digital goods within the tradicional goods/services distinction. Internation Journal of Law and Information Technology (Vol. 25, Is. 1)*, pp. 63-84.

Hollaar, L. A. (2002). *Legal Protection of Digital Information*. Obtenido de *Digital Law Online: Software Copyright:* http://digital-law-online.info/lpdi1.0/treatise17.html

Ifrah, G. (2001). *The universal history of computing: from the abacus to the quantum computer.* New York: John Wiley & Sons Inc. p. 11.

Jijena, R. (1996). *Copia ilegal de software y virus informáticos: un maridaje ilícito. Revista de Derecho de la Universidad Católica de Valparaíso (No. 17)*, pp. 313-330.

Jullien, B., & Sand-Zantman, W. (2016). *Network Effects. Institut D'Economie Industrielle (No. 27)*, pp. 3-7; 11-12; 19-23.

Kamantauskas, P. (2014). *Formation of Click-Wrap and Browse-Wrap Contracts. Lithuanian Academic Electronic Library*, pp. 7-8.

Lara, J. C., & Vera, F. (2014). *Medidas de tecnológicas de protección de propiedad intelectual: desafíos regulatorios en Chile. ONG Derechos Digitales (Policy Paper No. 01)*, pp. 3-4; 7-10.

Menell, P. S., Lemley, M. A., & Merges, R. P. (2019). *Intellectual Property in the New Technological Age: 2019.* Clause 8 Publishing, pp. 16-22; 29-30; 35-36; 43-47; 88-96.

Moglen, E. (2001). *Enforcing the GNU GPL*. Obtenido de *The GNU Proyect:* https://www.gnu.org/philosophy/enforcing-gpl.es.html

Moreno, R. D. (1990). *"Look and Feel" as A Copyrightable Element: The Legacy of Whelan v. Jaslow? Or, Can Equity in Computer Program Infringement Cases Be Found Instead By the Proper Allocation of Burden of Persuasion. Louisiana Law Review (Vol. 51, No. 1)*, pp. 177-211.

National Commission on New Technological Uses of Copyrighted Works. (1978). *Final Report of the National Commission on New Technological Uses of Copyrighted Works.* Washington D.C, pp. 23-33; 47-57; 66-93

Organización Mundial de la Propiedad Intelectual. (2007). *Digital Ecosystem.* pp. 1-6.

Organización Mundial de la Propiedad Intelectual. (2016). *Understanding Copyright and Related Rights.* pp. 3-19; 23-26.

Open Source Initiative. (2007 - 2017). *Open Source Initiative*. Obtenido de *Open Source Initiative:* https://opensource.org/

Perens, B. (1998). *The Open Source Definition*. Obtenido de *O'Reilly Media:* https://www.oreilly.com/openbook/opensources/book/perens.html

Peterson, S. K. (2018). *Why so little love for the patent grant in the MIT License?* Obtenido de *Opensource.com:* https://opensource.com/article/18/3/patent-grant-mit-license

Peterson, S. K. (2020). *Is open source licensing broken?* Obtenido de *Opensource.com:* https://opensource.com/article/20/2/open-source-licensing

Pianon, A. (2004). *Trade Secret Vs. Open Source: And the Winner Is. Erasmus Law and Economics Review 1*, 47-75.

Plana, J. (2016). *El agotamiento del derecho de distribución y su aplicación en un entorno digital. Revista Chilena de Derecho y Tecnología (Vol. 5, No. 2)*, 11-12; 17-20; 21-29; 35-38.

Posner, R. A. (2012). *Do patent and copyright law restrict competition and creativity excessively?* Obtenido de *The Becker-Posner Blog*: https://www.becker-posner-blog.com/2012/09/do-patent-and-copyright-law-restrict-competition-and-creativity-excessively-posner.html

Purewal, J. (2012). *The legality of second hand software sales in the EU*. Obtenido de *Gamer/Law:* http://www.gamerlaw.co.uk/2012/the-legality-of-second-hand-software-sales-in-the-eu/

R. Kerr, I., Maurushat, A., & Christian, S. T. (2005). *Technological Protection Measures: Tilting at Copyright's Windmill. Ottawa Law Review Vol. 34, No.1*, 13-22.

Ravicher, D. (2002). *Software Derivative Work: A Jurisdiction Dependent Determination*. Obtenido de *Linux.com:* https://www.linux.com/news/software-derivative-work-jurisdiction-dependent-determination/

Raymond, E. S. (2000). *A Brief Story of Hackerdom*. Obtenido de *Eric S. Raymond HomePage:* http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/

Raymond, E. S. (2000). *The Revenge of the Hackers*. Obtenido de *Eric S. Raymond HomePage:* http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-revenge/

Raymond, E. S. (2003). *The Art of Unix Programming.* Ch.1 Philosphy. Obtenido de *Eric S. Raymond HomePage: http://www.catb.org/~esr/writings/taoup/html/*

Red Hat Inc. (2020). *The State of Enterprise Open Source.* Obtenido de *RedHat:* https://www.redhat.com/cms/managed-files/rh-enterprise-open-source-report-detail-f21756-202002-a4-es.pdf

Rosen, L. (2004). *Open Source Licensing: Software Freedom and Intellectual Property Law.* Prentice Hall. pp. 103-107; 115-125.

Rosen, L., & Einschlag, M. B. (25 de mayo de 2004). *Derivative Works*. Obtenido de *Rosenlaw & Einschlag:* https://www.rosenlaw.com/lj19.htm

Samuelson, P. (2013). *A Fresh Look at Tests for Nonliteral Copyright Infringement. Northwestern University Law Review (Vol. 107, No. 4)*, pp. 1822-1823; 1837-1839; 1842-1843; 1847-1849.

Samuelson, P., Davis, R., Kapor, M. D., & Reichman, J. (1994). *A Manifesto Concerning The Legal Protection of Computer Programs. Columbia Law Review (Vol. 94, No. 8),* pp. 2315-2330; 2332-2342; 2371-2378; 2420-2426.

Smith, E. (2017). *The Incredibly Technical History of Digital Rights Management*. Obtenido de *Vice Media Group:* https://www.vice.com/en_us/article/evbgkn/the-incredibly-technical-history-of-digital-rights-management

Sono, H. (2008). *The Applicability and Non-Applicability of the CISG to Software Transactions. C. B. Andersen, U. G. Schroeter (eds.). Sharing International Commercial Law across National Boundaries: Festschrift for Albert H. Kritzer Wildy, Simmonds & Hill Publishing, London, 2008,* pp. 512-526. Obtenido de *Institute of International Commercial Law: https://www.cisg.law.pace.edu/cisg/biblio/sono6.html*

Stallman, R. (1998). *The GNU Operating System and the Free Software Movement: The Free Software Sharing Community*. Obtenido de *Dulce Libertad:* https://lorenzopena.es/linux/movement.html

Stallman, R. (2016-2018). *License Compatability and Relicensing*. Obtenido de *GNU Operating System:* https://www.gnu.org/licenses/license-compatibility.html

The Linux Information Project. (2006). *Software Definition*. Obtenido de *LINFO:* http://linfo.org/software.html

The Linux Information Project. (2005). *Algorithms: A Very Brief Introduction*. Obtenido de *LINFO:* http://www.linfo.org/algorithm.html

The Linux Information Project. (2006). *Files: A Brief Introduction*. Obtenido de *LINFO:* http://www.linfo.org/file.html

Ungericht, R. (2013). *From Calculators to Computers - A Brief History.* pp. 3-18.

Välimäki, M., & Oksanen, V. (2006). *DRM Interoperability and Intellectual Property Policy in Europe. European Intellectual Property Review (Vol. 26, No. 11)*, 562-568.