

DERRIBANDO ALGUNOS MITOS EN TORNO A LA INGENIERÍA INVERSA DE SOFTWARE

José Luis Baro Ríos¹

SUMARIO: INTRODUCCIÓN.- 1.- MARCO CONCEPTUAL.- 2. PRINCIPALES APLICACIONES DE LA DESCOMPILACIÓN.- 3. ANÁLISIS ECONÓMICO Y SOCIAL.- 4.- PROTECCIÓN JURÍDICA DEL SOFTWARE.- 5.- LA INGENIERÍA INVERSA DE SOFTWARE ES UNA ACTIVIDAD DE CARÁCTER NEUTRO.- 6.- RECONOCIMIENTO DE LA DESCOMPILACIÓN.- 7. LA INGENIERÍA INVERSA DE SOFTWARE COMO EXCEPCIÓN.- 8.- CONCLUSIONES.

INTRODUCCIÓN

Mucho se ha hablado acerca de la amenaza que el reconocimiento de la ingeniería inversa de software, como excepción o limitación a los Derechos de Autor, representa para los intereses de los derechohabientes. Se ha dicho que su establecimiento dentro del nuevo catálogo de excepciones y limitaciones que se pretende introducir en la actual revisión a nuestra legislación autoral podría transformar a nuestro país en un verdadero “paraíso de la piratería”. El poder de la retórica en el momento y lugar adecuados puede ser insuperable. Con todo, ni el autor del proyecto de reforma, ni los detractores de la descompilación en nuestro país, se han detenido a precisar en qué consiste la ciencia que pasará a ocupar las líneas que siguen del presente trabajo. Por ello y con la esperanza de que el mismo constituya un aporte oportuno a la discusión, intentaré hacerme cargo de dicha tarea.

Sólo sabiendo exactamente qué es la ingeniería inversa de software o descompilación y conociendo sus aplicaciones prácticas y beneficios para el avance de las ciencias y de la innovación y desarrollo en nuestra economía, podremos saber si la misma es pro-

¹ José Luis Baro Ríos es abogado por la Universidad de Chile. Actualmente, integra el estudio jurídico García Magliona & Compañía Abogados, en el área de propiedad intelectual, nuevas tecnologías y derecho del entretenimiento. Durante los últimos años se ha dedicado a la investigación de la regulación de los derechos de propiedad intelectual sobre obras y prestaciones protegidas en el entorno digital, en general, y al tratamiento jurídico del software, en particular.

cedente desde el punto de vista de los principios generales que informan el Derecho de Autor, tales como la exigencia de originalidad, cuya contrapartida es la exigencia de que la obra en cuestión no se encuentre en el dominio público y, en especial, respecto de la dicotomía idea vs. expresión. Del mismo modo, podremos determinar si esta disciplina resultará efectivamente perjudicial para los intereses de los titulares de los derechos sobre los programas descompilados, teniendo presente el gran nivel de protección de que estas obras cuentan a partir de la última revisión a la Ley N° 17.336 del año 2003.

1.- MARCO CONCEPTUAL

La ingeniería inversa constituye una disciplina de larga data y que ha sido durante mucho tiempo una herramienta puesta a disposición del progreso tecnológico. Podemos definir la ingeniería inversa como “el proceso encaminado a obtener, a partir de toda creación humana, (...) fragmentos perdidos de conocimiento, ideas y filosofía de diseño cuando esa información no se encuentra disponible, ya sea porque su propietario no desea compartirla o porque la información en sí misma, se ha extraviado o destruido”. En otros términos, el proceso a través del cual un artefacto diseñado a partir de un método ingenieril (como un automóvil, la turbina de un avión o un programa computacional) es descompuesto, de una forma que revela sus detalles más internos, tales como su diseño y arquitectura.²

Si todo proceso ingenieril encaminado a la producción de un bien intelectual se inicia con la concepción de una idea en la mente del creador, seguida por la definición de los modelos o técnicas a seguir, diseño y desarrollo de prototipos para culminar con la invención o creación determinada, entendida como expresión única de la idea inicialmente concebida, podemos afirmar que la ingeniería inversa sigue el camino contrario: tomando el bien intelectual acabado, se procede a su descomposición o desensamblaje, siguiendo un camino en reversa para llegar a conocer los detalles internos de su diseño, la forma como opera hasta llegar a la aprehensión de la idea misma en los niveles o capas mayores de abstracción.³

La importancia de la ingeniería inversa radica en que ella permite a los investigadores avanzar en aquellos campos en los cuales la innovación es acumulativa. Es por ello que se ha sostenido que la ingeniería inversa constituye “una parte esencial de la innovación” capaz de producir variaciones en un producto “que pueden conducir a avances significativos en la tecnología”.⁴

2 EILAM Eldad, “Reversing: secrets of reverse engineering”, Wiley Publishing Inc. Indianapolis, Indiana, 2005, p. xxiv.

3 En el mismo sentido, vid. Electronic Frontier Foundation - Brief of Amicus Curiae on Behalf of IEEE-USA, in DVD-CCA v. Bunner CA; July 5, 2000. [en línea] <http://www.eff.org/IP/Video/DVDCCA_case/20000705_def_ieee_appeal_amicus.html#position> [consulta: 15 de septiembre de 2007]

4 Ver sentencia Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141 (1989) [en línea] <http://caselaw.lp.findlaw.com/scripts/getcase.pl?navby=search&friend=nytimes&court=US&case=/us/489/1_41.html> [consulta:

Por su parte, la ingeniería inversa de software puede ser definida como “el proceso consistente en analizar un sistema dado, con el objeto de identificar los componentes de dicho sistema y sus interrelaciones y crear representaciones del mismo en otra forma o en un nivel de abstracción mayor”.⁵ Este proceso permite visualizar la estructura del software, las formas en que opera, y las características que manejan su comportamiento.

En los aparatos e invenciones de carácter tangible, el proceso de ingeniería inversa se inicia con la descomposición o desensamblaje del objeto que se pretende estudiar, a fin de entender su estructura general: de qué está hecho y qué lo hace funcionar, información que podrá ser aplicada para entender y mejorar la tecnología en él contenida en futuros proyectos. Por ello se dice que en la industria manufacturera tradicional, esta disciplina es empleada principalmente para el desarrollo de nuevos productos, innovadores y competitivos. En el caso del software esta disciplina se diferencia de la regla general antes descrita, en dos aspectos: la forma o método que se emplea para ponerla en práctica y el objetivo principal al cual se apunta por intermedio suyo.

La diferenciación en la forma o método empleado para practicar ingeniería inversa a un programa computacional, viene dada por la estructura particular de este tipo de obras, esto es, por los lenguajes en que las mismas se encuentran expresadas en sus distintas fases de desarrollo. Hablamos, en este sentido, de software expresado en lenguaje de alto nivel o código fuente y de software expresado en lenguaje de máquina, o código objeto. La ingeniería inversa de software consta fundamentalmente de tres fases: descompilación, análisis y estudio, y aplicación de las ideas extraídas.

La descompilación es el proceso a través del cual se procede a traducir el código objeto del programa a un lenguaje susceptible de ser leído y entendido por los seres humanos, esto es código fuente o lenguaje de programación. El proceso de descompilación no genera una versión de dicho código como originalmente fue escrito, sino más bien una “reconstrucción plausible” de porciones del código original, de la forma como pudo haberlo expresado su creador.⁶ En este punto, cabe hacer presente que la descompilación requiere que el programa descompuesto sea al menos copiado una vez en el disco del sistema empleado para llevar a cabo el proyecto.

El estudio y análisis, a partir de la “aproximación” al código fuente, obtenida durante el proceso de descompilación, el investigador puede discernir o deducir los detalles internos del diseño del programa, los elementos que le dan su funcionalidad

09 de septiembre de 2007]

5 E. J. CHIKOFSKY y J. H. CROSS, “Reverse Engineering and Design Recovery: a Taxonomy”, IEEE Software, 7:13-17, January 1990, en Eldad EILAM, Op. Cit., p. viii.

6 Julie E. COHEN and Mark A. LEMLEY, “Patent Scope and Innovation in the Software Industry”, 89 Calif. L. Rev. 1, 6 (2000), en Pamela SAMUELSON y Suzanne SCOTCHMER, “The Law & Economics of Reverse Engineering” [en línea] <<http://people.ischool.berkeley.edu/~pam/papers/l&e%20reveng3.pdf>> [consulta 09 de septiembre de 2007], p. 28.

única, así como aquellos que permiten la obtención de la compatibilidad operativa con otro dispositivo, aplicación o plataforma.

Finalmente, la aplicación de los conocimientos e ideas extraídas a partir del estudio del código fuente. El conocimiento adquirido durante el estudio de los fragmentos de código fuente obtenidos a partir de la descompilación puede ser empleado para diversos fines, tales como evaluación de la calidad del programa, investigaciones en criptografía o en el desarrollo de nuevos programas compatibles o competitivos.

La aplicación de técnicas de ingeniería inversa a un software, toma sentido en atención a la licencia con que su creador lo distribuye.⁷ En este sentido, existen una serie de programas cuyo código fuente se encuentra disponible al público, lo cual hace innecesario el proceso de descompilación para conocer su diseño, características y funcionalidad. Así, encontramos el llamado software libre (free software), que es aquel que puede ser usado, estudiado, modificado, copiado y redistribuido en su versión original o modificada, en uno u otro caso, sin restricciones, o sólo con las limitaciones necesarias para asegurar que los usuarios finales gocen de la misma libertad que los creadores originales. De otra parte el llamado software de código abierto (open source software) comparte con el software libre la última característica, esto es, el tener su código fuente disponible para los usuarios.

Sin embargo, la mayor parte del software en circulación obedece a la categoría opuesta a las antes descritas: el llamado software no libre, privado o privativo (“proprietary software”), en el cual el código fuente no se encuentra disponible, resultando necesario obtener autorización expresa del titular de los derechos sobre el mismo, para estudiarlo y trabajar sobre él. Es sólo respecto de esta categoría de software que la aplicación de técnicas de ingeniería inversa toma sentido.

2.- PRINCIPALES APLICACIONES DE LA DESCOMPILACIÓN

Podemos dividir las aplicaciones de la descompilación en dos grandes apartados: seguridad y desarrollo.

En sede de seguridad la descompilación juega un rol importantísimo en el estudio y desarrollo de mecanismos de defensa contra el software malicioso o malware. Es, de hecho, la técnica utilizada por los fabricantes de antivirus para actualizar las bases de datos de sus programas y actualizarlos en beneficio de sus clientes.

En el mismo apartado, la descompilación tiene gran aplicación en el terreno de los estudios de codificación y encriptado. El objetivo de la encriptación en el entorno

⁷ Mark LEMLEY, Peter MENELL, Robert MERGES y Pamela SAMUELSON, “Software and Internet Law”, 3th Ed., Aspen Publishers. NY. U.S., 2006, p. 299.

numérico es hacer que el acceso a la obra codificada esté reservado sólo para quien posea una determinada clave o llave (key). Esta llave es conocida como algoritmo criptográfico (cryptographical algorithm, también conocido como cipher) que no es más que una función matemática utilizada para encriptar y desencriptar. Por medio de ingeniería inversa aplicada al software, puede determinarse el nivel de seguridad proporcionado por un determinado sistema de acceso condicionado. En consecuencia toda actividad de investigación en materia de encriptación es, en sí misma, una actividad de ingeniería inversa.

Las llamadas medidas tecnológicas de protección, se basan fundamentalmente en las tecnologías de codificación y encriptado. Ellas constituyen mecanismos insertos sobre obras y prestaciones protegidas por el Derecho de Autor, por parte de los derechohabientes y que están destinadas a controlar el acceso a las mismas, así como los usos que se pueden realizar de ellas. Aunque las mismas son aplicables a toda clase de obras y prestaciones protegidas, nos referiremos a las que se encuentran insertas en obras expresadas en lenguaje numérico, esto es, en formato digital.

Las medidas tecnológicas de protección pueden agruparse en dos grandes grupos: medidas tecnológicas de control de acceso y medidas tecnológicas de control de uso.

Las primeras, también conocidas como medidas de acceso condicionado, son aquellas que impiden el acceso a la obra si el usuario no cumple con la condición impuesta por el titular de los derechos sobre la misma, principalmente, el pago de la remuneración debida por la adquisición del soporte que la contiene o su descarga desde un servidor determinado. Por ejemplo, en el ámbito de las obras cinematográficas, pueden citarse las divisiones por áreas geográficas asignadas a los DVD que impiden su reproducción en un país diferente de aquel en el que se ha adquirido el soporte respectivo.

El segundo grupo, esto es las medidas tecnológicas de control de uso, corresponden a todos aquellos mecanismos insertos en una obra y que limitan los usos y actividades que pueden realizar los usuarios de la misma, en relación con la facultad exclusiva de explotación garantida por la ley al titular de los derechos de autor, esto es, la reproducción, comunicación, publicación o puesta a disposición del público. Dentro de este segundo grupo se incluyen, asimismo, las medidas destinadas a identificar las obras y prestaciones a fin de rastrear su utilización (CMI) y las destinadas a posibilitar la gestión electrónica de los derechos (DRM).

La ingeniería inversa de software, mirada desde la perspectiva de las MTP reviste una importancia tremenda, tanto para los titulares del derecho de autor, como para la comunidad de usuarios de este tipo de obras. Para los primeros, la ingeniería inversa constituye una herramienta de gran utilidad para las investigaciones de codificación y encriptado, y de este modo sirven para detectar las vulnerabilidades o brechas de

seguridad y buscar la solución para hacerlo más seguro y robusto. Mientras que, desde el punto de vista del usuario legítimo de este tipo de obras, de quien ha adquirido el soporte por medios válidos, en ocasiones las MTP pueden impedir un uso de la obra de una manera para la cual el primero se encuentra autorizado, ya sea en virtud de una excepción o limitación expresamente contemplada en la legislación autoral o, bien, por estimarse que el uso que se pretende hacer de la obra es de carácter legítimo o razonable, en aquellos sistemas donde la doctrina del fair use resulta aplicable. Aquí la ingeniería inversa es útil para restablecer la vigencia de tales legítimos usos.

Siguiendo en el terreno de la seguridad, la ingeniería inversa sirve para el estudio de la calidad del código de un software privado o privativo por medio de la auditoría de sus binarios. Finalmente, puede detectarse mediante la descompilación el empleo del código ajeno, por parte de un tercero, en el marco de una pericia realizada para detectar una infracción al derecho de autor por haberse incurrido en la copia de los elementos literales del programa, al desarrollar un programa independiente.

Pasaremos ahora a analizar las mayores aplicaciones de esta ciencia en el apartado de desarrollo de software independiente: la compatibilidad operativa y el desarrollo de programas competitivos.

La interoperabilidad es el terreno donde los programadores e informáticos en general pueden verse mayormente beneficiados por la aplicación de la ingeniería inversa. Consiste en el desarrollo de un programa que se comuniquen con un componente desarrollado por otra compañía. Si un programador desea desarrollar un programa para una determinada plataforma, debe tener acceso a especificaciones muy precisas sobre la forma en la cual los servicios brindados por la misma pueden ser solicitados y obtenidos.

El desarrollador de una nueva plataforma puede optar por hacerla compatible, publicando sus interfaces o bien, ponerlas a disposición del público mediante licencias de open source software. En ambos casos, la aplicación de ingeniería inversa, resulta innecesaria para que los desarrolladores puedan adaptar sus programas o bien desarrollar nuevas aplicaciones para la plataforma en cuestión.

El desarrollador de una plataforma puede optar, sin embargo, por la vía contraria, esto es, restringir el acceso a sus interfaces, extendiendo licencias restringidas a desarrolladores independientes.⁸ ¿Qué sucede en aquellos casos en los cuales una compañía independiente decide desarrollar por sí una aplicación compatible con una determinada plataforma no libre, privada o privativa, sin adquirir licencia alguna? Esto, que en principio puede parecer ilegal, especialmente a la luz de la protección brindada a los

8 M.M. MOORE, "A license to practice software engineering", Software IEEE, Vol. 20, Año 2003 {en línea} <http://ieeexplore.ieee.org/xpl/freabs_all.jsp?tp=&arnumber=1196335&isnumber=26915> {consulta: 11 de septiembre de 2007}.

programas computacionales por las leyes de propiedad intelectual, las licencias de desarrollo extendidas por el fabricante y la concesión de patentes que resguardan este tipo de creaciones en algunas legislaciones, es una cuestión que se ha suscitado en la práctica, especialmente por la concurrencia de dos factores: los efectos globales asociados al mercado de algunas plataformas informáticas y el contenido de las licencias extendidas por el fabricante.

Ciertas plataformas, como los sistemas operativos, se caracterizan por encontrarse insertas en un mercado que tiende hacia la estandarización alrededor de un solo producto, en atención a que los consumidores se benefician al trabajar en la misma plataforma que todo el mundo utiliza. Esta situación confiere, en el hecho, a los fabricantes de determinados sistemas operativos una posición dominante en el mercado.⁹ El titular de los derechos sobre una plataforma dominante goza, asimismo, de un amplio margen para imponer las condiciones bajo las cuales extenderá una licencia para el desarrollo de programas compatibles con ella. Al optar por el desarrollo de una plataforma no libre, el dueño adquiere un poder considerable sobre las aplicaciones que funcionarán en ella.

El mercado potencial del desarrollador independiente se ve, en esos términos, considerablemente coartado. Es un tema de estudio de costo/beneficio y el interés de los desarrolladores independientes por proceder a la descompilación para no verse atados por dichos términos y condiciones desfavorables, quedó de manifiesto en el caso *Sega vs Accolade*, de principios de los años noventa en los EEUU.

Ahora, distinto es el caso respecto del empleo de la descompilación para el desarrollo de programas nuevos que compitan con el producto referencial, descompilado o estudiado. La práctica de ingeniería inversa sobre la totalidad de un programa, para el desarrollo de otro más competitivo, no es rentable. El proceso de descompilación demanda el consumo de mucho tiempo y recursos. De no contarse con las herramientas ni la experticia adecuadas, las posibilidades de éxito son inciertas.

En consecuencia y a diferencia de lo que ocurre en materia de desarrollo de aplicaciones compatibles, donde la ingeniería inversa ha demostrado ser una herramienta muy eficiente colocada a disposición de los desarrolladores independientes, en materia de programas competitivos es mucho más sustentable desde el punto de vista financiero, optar por el desarrollo de un nuevo código desde cero o bien licenciar los componentes más complejos de la tecnología empleada por terceros desarrolladores.¹⁰

9 Peter S. MENELL, "Tailoring Legal Protection of Computer Software", en Mark LEMLEY, Peter MENELL, Robert MERGES y Pamela SAMUELSON, Op. Cit., p. 289.

10 Pamela SAMUELSON y Suzanne SCOTCHMER, Op. Cit., p. 32.

3.- ANÁLISIS ECONÓMICO Y SOCIAL

Pamela Samuelson y Suzzane Scotchmer, ponderan los beneficios económicos de esta actividad, en el terreno de la elaboración de programas compatibles, en torno a cuatro criterios de bienes sociales: incentivo al desarrollo de plataformas no compatibles, incentivo al desarrollo de aplicaciones, nivel de los precios y nivel de gastos innecesarios en la producción de software.¹¹

En primer lugar, el incentivo al desarrollo de plataformas no compatibles disminuye. Si los desarrolladores pueden acceder a la plataforma gracias a la ingeniería inversa, el poder del dueño de la misma sobre el mercado disminuye. Como consecuencia de ello, se verá forzado a extender licencias en términos más favorables para los terceros desarrolladores. Con todo, los perjuicios patrimoniales que potencialmente pueden sufrir los titulares de los derechos sobre la plataforma se ven atenuados por dos factores: a) el carácter lento y costoso de la ingeniería inversa de software, y, b) El nivel de entrada de los terceros en el mercado.

En segundo lugar, el incentivo al desarrollo de aplicaciones aumenta, desde que el acceso a las interfaces de la plataforma, posibilitado gracias a la ingeniería inversa, permite el ingreso al mercado de un gran número de programas creados por desarrolladores independientes. La apertura de las interfaces permite a terceros desarrolladores ingresar al mercado sólo en el nivel de las aplicaciones, en lugar de verse constreñidos a ingresar en el de las plataformas y las aplicaciones. Además, la ingeniería inversa no sólo incentiva a terceros desarrolladores a crear nuevas aplicaciones, sino también adaptar las existentes a múltiples plataformas.

En tercer lugar, los efectos de la ingeniería inversa sobre el precio de las aplicaciones (factor que constituye el principal impacto sobre la comunidad desde que se traduce en la posibilidad de acceder a este tipo de obras, por parte de los diversos sectores de la sociedad), es un tema discutido. Por regla general, cuando la ingeniería inversa es procedente, en un escenario tradicional manufacturero, el nivel de los precios tiende a la baja, al introducirse mayor competencia en el mercado. Con todo, en la materia que estamos tratando, la tendencia, según algunos comentaristas, es la contraria: los precios son más altos cuando los sistemas son compatibles, que cuando no lo son¹², fundamentalmente por dos razones: a) la competencia entre plataformas no compatibles es más fuerte que

11 *Ibidem*, Op. Cit., pp. 37 y ss.

12 Ver Douglas LICHTMAN, "Property Rights in Emerging Platform Technologies", 29 *J. Legal Stud.* 615 (2000); Julie E. COHEN and Mark A. LEMLEY, "Patent Scope and Innovation in the Software Industry", 89 *Calif. L. Rev.* 1, 6 (2000); Carmen MATUTES and Pierre REGIBEAU, "Mix and Match: Product Compatibility Without Network Externalities", 19 *RAND J. Econ.* 221 (1988); Jeffrey CHURCH and Neil GANDAL, "Systems Competition, Vertical Merger and Foreclosure", 9 *J. Econ. & Mgmt. Strategy* 25 (2000); Augustin COURNOT, "Researches into the Mathematical Principles of the Theory of Wealth" (1838); Pamela SAMUELSON y Suzanne SCOTCHMER. Op. Cit., p. 39.

entre aquellas que lo son, lo cual conduce a una baja en los precios; b) si una sola compañía vende piezas complementarias como un todo, lo hará a un precio total más bajo que dos compañías vendiendo los componentes por separado, beneficiando tanto a los consumidores como a la propia firma.

Por otro lado, se sostiene que la inexistencia de plataformas compatibles puede traducirse en bajos precios, en el corto plazo, pero en el mediano y largo plazo puede llegarse a una alza considerable de los mismos. La incompatibilidad puede llevar a una guerra de precios en pos de alcanzar la estandarización, dejando en definitiva una sola plataforma en el mercado. Sin embargo, con interfaces privativas inmunes a la ingeniería inversa, los consumidores pueden encontrarse con precios bajos en un principio, pero más altos, en el mediano y largo plazo, tras la victoria de un solo sistema en la “guerra por la estandarización” y la desaparición de la presión por mantener el nivel de los precios.¹³

Finalmente, en cuarto lugar, en cuanto a la duplicación o gastos innecesarios en la producción de nuevos programas, la misma puede producirse, en este contexto, en tres actividades: la ingeniería inversa, en sí misma; el desarrollo de medidas tecnológicas de protección que impidan la descompilación del código de la plataforma, y en el desarrollo de diferentes aplicaciones para diferentes interfaces en lugar de una sola aplicación multiplataforma. De prohibirse la ingeniería inversa, podrían evitarse las dos primeras, pero la duplicación de gastos se produciría en el desarrollo de distintos programas con igual funcionalidad para distintas plataformas. A contrario sensu, de legitimarse la descompilación, podrían desarrollarse programas multiplataforma y evitar la duplicación de gastos en este apartado, no así en las dos primeras actividades antes citadas.

Como se puede apreciar, en la teoría, el reconocimiento de la ingeniería inversa de software para la obtención de compatibilidad operativa puede acarrear, a nivel económico y social, efectos eclécticos. Con todo, ponderando todos los factores antes expuestos, su legitimación resultaría beneficiosa para la comunidad en su conjunto, desde que se incentivaría ampliamente el desarrollo y distribución de nuevos programas multiplataforma y de gran calidad.

4.- PROTECCIÓN JURÍDICA DE LOS DERECHOS SOBRE EL SOFTWARE EN LA LEGISLACIÓN AUTORAL: ALCANCE DEL DERECHO EXCLUSIVO DE REPRODUCCIÓN

En nuestro sistema autoral, a los programadores, esto es, los autores del software, no cabe sino reconocérseles todo el catálogo de derechos patrimoniales, esto es, de reproducción, publicación, adaptación y ejecución o comunicación pública, y morales, como el de reivindicar la paternidad de la obra, el de protección de su integridad e

13 Ibid.

incluso el derecho al inédito. En el caso de los publishers y empresas desarrolladoras de software, en su calidad de titulares derivados del Derecho de Autor, les debemos reconocer toda la gama de derechos patrimoniales de autor. Con todo, en el presente apartado, nos referiremos al alcance del derecho exclusivo de reproducción, en relación a las infracciones que se pueden suscitar en el marco del desarrollo de programas por parte de terceros o, dicho de otra manera, cuáles son los elementos que se estiman protegidos en un programa computacional y por ende no se pueden emplear en el marco del desarrollo de nuevas obras.

No nos referiremos a la “clonación de las obras”, entendida como la reproducción exacta y no autorizada ni amparada en la excepción contemplada en la ley de propiedad intelectual, respecto de un programa computacional expresado en lenguaje de máquina en un soporte idéntico al matriz, conducta que puede ser fácilmente subsumida en las infracciones contempladas en el citado cuerpo legal, ya sea considerada como reproducción no autorizada o como piratería, cuando la misma es realizada en un contexto lucrativo.

El estatuto jurídico de los programas computacionales, como lo conocemos hoy en día ha sido producto de un intenso debate que se inició desde el nacimiento de la industria informática hace casi 50 años. Al decantarse el concepto de programa computacional, como “conjunto de instrucciones destinadas a ser ejecutadas por un ordenador para el logro de fines específicos”, se estimó que dicho “conjunto de instrucciones” era lo que mejor calzaba para describir este tipo de obras, como bienes intelectuales, asimilándoselos a las “obras literarias”. Sin embargo, el carácter utilitario de los mismos ha traído como consecuencia que, hasta el día de hoy, los derechohabientes busquen su protección bajo el sistema de patentes (situación aceptada por los servicios respectivos, en los Estados Unidos) e incluso en la institución del secreto de fabricación. Algunos sistemas, como el brasileño, aceptan un tratamiento jurídico sui generis para este tipo de obras.

Hoy en día el régimen de protección autoral de este tipo de obras es aceptado de manera casi universal.¹⁴ En consecuencia la protección brindada a los programas computacionales alcanza tanto a sus elementos literales, esto es, el texto mismo como a sus elementos “no literales”, esto es, aquello que sin estar en el texto mismo, fluye de la obra como su apariencia, la sensación que provoca en el usuario y que reúne la exigencia de originalidad suficiente para ser considerado como expresión de una idea, digna de protección ¿Cómo llevamos estos principios al terreno del software?

La Ley N° 17.336 establece que quedan especialmente protegidos 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

14 Vid. Artículo 4 del Tratado OMPI sobre Derecho de Autor de 1996.

manuales de uso. De esta manera, nuestra legislación autoral protege los derechos de los autores y titulares del derecho de autor en cuanto recaen sobre los siguientes elementos del software:

- Se reconoce la protección del programa ya sea se encuentre expresado en lenguaje de programación (código fuente) o de máquina (código objeto). En uno u otro caso, estamos hablando de los elementos literales del programa computacional. La copia de los mismos se da cuando el infractor toma directamente el texto de la obra, en el caso del programa, no su organización o resultado, sino el código fuente u objeto en sí. No hablamos de “similaridad substancial”, sino del empleo directo de estos elementos.¹⁵

- Los accesorios del programa computacional: en primer lugar, la documentación preparatoria, la documentación técnica y el manual del usuario.

La protección brindada por la legislación autoral a los programas computacionales se extiende más allá de sus elementos literales, cubriendo lo que se conoce como los “elementos no literales del programa”, que son aquellos que integran la “estructura general” (secuencia y organización) del programa y que surgen mucho antes de la etapa de codificación, cuando el programador comienza a trazar una vía de solución para el problema planteado, esto es, al concebir los módulos, subrutinas y estructuras de archivos del futuro programa y combinarlos entre sí. Ello precisamente, porque la mayor labor creativa del programador se da frecuentemente durante la etapa de diseño, más que en la codificación.¹⁶ Habrá copia de los elementos no literales de un programa computacional, cada vez que exista similaridad substancial entre dos obras sujetas a examen.

De esta manera, los elementos no literales de un programa computacional, pueden ser definidos como “aquellos elementos diferentes del código escrito, tales como los diagramas de flujo (que proporcionan la estructura detallada de la cual deriva el

15 En la actualidad, no cabe duda alguna que la legislación autoral protege la totalidad de los elementos literales de los programas computacionales. La mención expresa de los mismos en el actual numeral 16) del artículo 3º, como “programa fuente” o “programa objeto”, disipa toda duda que en lo pertinente pudiera alzarse. Con todo, en los primeros años de discusión sobre el tema, se planteaba la interrogante de si el código objeto podía ser de alguna forma protegido por el derecho de autor o si dicho amparo se encontraba circunscrito sólo al código fuente, ya que la protección legal de las creaciones “utilitarias”, se encuentra reservado a las leyes sobre patentes. Hasta la década de los 60s - 70s, habían sido protegidas por medio del secreto de fabricación. Ver Mark LEMLEY, Peter MENELL, Robert MERGES y Pamela SAMUELSON, *Op. Cit.*, pp. 4 y 35. Al establecerse su asimilación a las “obras literarias”, comienza a discutirse acerca de si procedía distinguir entre el código fuente y el código objeto, para los efectos de determinar el nivel de protección de que cada uno gozaba. Esta materia fue discutida en los Estados Unidos a fines de la década de los 1970s en un litigio que se extendió hasta el mes de agosto del año 1983. El caso *Apple Computer, Inc. v. Franklin Computer Corporation*, sentó un precedente fundamental para la determinación del nivel de protección de que gozarían los elementos literales de los programas computacionales. En definitiva, la Corte estableció que la copia exacta del programa en código objeto era infracción a las leyes sobre copyright, basado en que la idea del sistema operativo Apple era hacer funcionar un computador de dicha compañía y que la “forma particular como Apple hace funcionar dicho aparato” es una expresión protegida.

16 Ver sentencia *Whelan Associates Inc. v. Jaslow Dental Laboratory, Inc.* Considerando III. [en línea] <<http://digital-law-online.info/cases/230PQ481.htm>> [consulta: 24 de marzo de 2008]

código escrito), listas de parámetros (piezas de información que son intercambiadas por las varias subrutinas generadas por el software) y la organización específica de los módulos del programa”.¹⁷

El objetivo del Derecho de Autor es brindar al creador de una obra protegida, un monopolio limitado temporalmente sobre la misma, con el objeto de explotarla y percibir la remuneración debida por su uso, como compensación a su labor creativa. Pero dicho monopolio se extiende a la “expresión”, no a la “idea que subyace a dicha expresión”, la cual no es objeto de protección, por lo cual, toda persona se encuentra facultada para aprehenderla.¹⁸

De esta manera, se debe tener especial cuidado en este terreno: el hecho de que el legislador nacional haya decidido extender la protección de los programas a la documentación preparatoria y técnica, no significa que todos los elementos consignados en ella deban estimarse protegidos: extender la protección autoral a las interrelaciones y funcionalidades obtenidas en un nivel abstracto, sin analizar el contenido del documento de especificaciones o de diseño, puede llevar al intérprete a extender artificialmente al monopolio garantido al titular de los derechos sobre el software a fórmulas, conceptos y métodos de operación que son de uso común, con los consecuentes perjuicios para el mercado y los terceros desarrolladores.

5.- LA INGENIERÍA INVERSA DE SOFTWARE ES UNA ACTIVIDAD DE CARÁCTER NEUTRO

Ya hemos analizado las principales aplicaciones de la descompilación y los beneficios que de ella emanan tanto para los titulares de los derechos sobre estas obras, como para la comunidad representada por usuarios y terceros desarrolladores deseosos de introducir nuevos productos en el mercado. Toca ahora hacernos cargo del más grande de los fantasmas que rodean a esta ciencia y que constituye una de las interrogantes planteadas al inicio del presente trabajo: si el reconocimiento jurídico de la ingeniería inversa de software puede, efectivamente, resultar perjudicial a los intereses de los derechohabientes, a la luz de la protección brindada a estas obras en la legislación autoral.

17 Ver sentencia *Computer Associates International, Inc. v. Altai, Inc.*, Nos. 91-7893, 91-7935, 1992 U. S. App. LEXIS 14305 (2d Cir. June 22, 1992). [en línea] <<http://links.jstor.org/sici?sici=0017-811X%28199212%29106%3A2%3C510%3ACLSOPO%3E2.0.CO%3B2-J&size=LARGE&origin=JSTOR-enlargePage>> [consulta: 09 de octubre de 2007]

18 La distinción “idea / expresión” en materia de Derecho de Autor, se encuentra recogida en el Acuerdo sobre los ADPIC, en su artículo 9.2.: Dicha distinción aparece consagrada en idénticos términos en el artículo 2 del Tratado OMPI sobre Derecho de Autor, de 1996. Haciendo un paralelo con el sistema de propiedad industrial, podemos observar que generalmente tampoco son patentables los contenidos ideativos generales, tales como métodos, principios o planes, o de simple verificación y fiscalización; y los referidos a actividades puramente mentales o intelectuales. Ver Christian SCHMITZ V., “Propiedad Industrial y Derecho de Autor ¿Una División Vigente?”, en “Temas Actuales de Propiedad Intelectual. Estudios en honor a la memoria del profesor Santiago Larraguibel Zavala”, Ed. Jurídica Lexis-Nexis, Santiago, 2006, p. 45.

Decimos que la ingeniería inversa de software presenta un carácter neutro en cuanto a su contenido ético–moral. Como toda actividad relacionada con la tecnología, constituye una herramienta que puede ser utilizada con fines lícitos o antijurídicos. Esta actividad no puede ser declarada como infractora de las leyes de propiedad intelectual, per se. Es el fin, el resultado perseguido por quien inicia un proceso de descompilación el que debe ser sujeto a examen.¹⁹

La filosofía que subyace a la ingeniería inversa es la misma: todo bien intelectual lleva consigo una serie de conceptos, ideas y conocimientos a los cuales la comunidad tiene el legítimo derecho de acceder. Los usuarios finales se ven beneficiados, desde que esta ciencia les permite realizar usos legítimos y razonables de las obras que han adquirido, de manera amplia, sin las limitaciones a que se encuentran sujetas en mérito de sus soportes originales, todo en un contexto no lucrativo, realizado por aficionados. Asimismo, el conocimiento adquirido beneficia al investigador, quien podrá aplicar dichos conceptos e ideas en futuros proyectos ya sea en el apartado seguridad o desarrollo. Nuevos actores podrán ingresar en el mercado con productos de calidad, competitivos o multiplataforma, innovando en relación a la tecnología existente, en beneficio de la comunidad.

Con todo, la integración única de los conceptos e ideas que subyacen a un programa computacional, en calidad de obra literaria reconocida por las leyes de Derecho de Autor, debe ser respetada: ella constituye la “expresión protegida” amparada por la legislación autoral. Es por ello que la ingeniería inversa, actividad de carácter neutro como se expuso anteriormente, debe ser empleada en un contexto legítimo. La línea que divide la simple copia, de la innovación, es muy delgada y por lo tanto, de producirse controversias en su aplicación, el Tribunal llamado a dirimir del conflicto debe hacer un análisis objetivo y minucioso de los antecedentes de hecho, en cada caso particular.

Desde este punto de vista, concluimos que el reconocimiento jurídico de la ingeniería inversa de software para fines de compatibilidad operativa, investigación o desarrollo, no dejará en el desamparo a los autores o titulares de los derechos sobre el software: el resultado final de la descompilación, el producto del proceso de desarrollo en el cual la misma se encuentra inserta, podrá siempre ser objeto de juicio, no pudiendo establecerse la ilicitud de las labores de ingeniería inversa, ex ante.

6.- RECONOCIMIENTO DE LA DESCOMPILACIÓN

Como se ha señalado con anterioridad, la descompilación de software, en el terreno del desarrollo de programas computacionales nuevos, independientes del producto referencial, tiene por principal aplicación, la obtención de aquellos elementos que

19 James POOLEY, “Trade Secret Law”. sec. 5-18 to 5-19 (1999), en Pamela SAMUELSON – Suzanne SCOTCHMER, Op. Cit., p. 5.

permiten alcanzar la compatibilidad operativa de una aplicación independiente, con una plataforma informática propietaria, esto es, el obtener el acceso a las interfaces de aplicación. Es sobre este punto donde la discusión doctrinal y más adelante, legislativa, en el derecho comparado, comenzó a gestarse hace casi veinte años y que al día de hoy ha tenido sus frutos en las distintas legislaciones que han asumido la tarea de regular esta importante materia.

El primer atisbo de discusión acabada del tema, a nivel jurisprudencial, lo encontramos en los Estados Unidos. La ingeniería inversa para fines de obtención de compatibilidad operativa entre programas desarrollados independientemente, fue en un comienzo validada por la jurisprudencia norteamericana,²⁰ y posteriormente pasó a constituir una excepción o limitación al copyright en la actual Sec. 1201(f) de la USCA, así como en las diversas legislaciones autorales en el resto del mundo.

Por su parte, la Comisión de la Comunidad Europea de Naciones, innovó el año 1991 con la adopción de la Directiva 91/250/CEE del Consejo del Parlamento Europeo, de 14 de mayo de 1991, sobre la protección jurídica de programas de ordenador. Tras largas discusiones, los Estados miembros acordaron incluir dos excepciones, permitiendo a los usuarios copiar un programa computacional con el fin de determinar las ideas y principios que subyacen a cualquier elemento del programa. En primer lugar, se estableció el derecho de los usuarios a realizar pruebas de “black box”, para fines de investigación.²¹ Y, de otra parte, la Directiva reconoció a los usuarios el derecho a descompilar el programa para fines de desarrollo de programas compatibles, estableciendo que no se requerirá autorización del derechohabiente cuando la copia del código o la adaptación del programa resulten indispensables para obtener la información necesaria para alcanzar la compatibilidad operativa de un programa computacional creado independientemente, con otros programas.²²

Pese a que en la última revisión a la legislación autoral en el concierto comunitario, en el marco de la Directiva sobre Derechos de Autor en la Sociedad de la Información del año 2001, no existe mención expresa a la descompilación, la excepción de ingeniería inversa para la obtención de compatibilidad operativa, adoptada a partir del año 1991, pervive en las legislaciones nacionales de prácticamente la totalidad de los Estados miembros de la Unión Europea.²³

20 Ver Sentencia *Sega Enterprises Ltd. v. Accolade, Inc.* United Status Court of Appeals for the Ninth Circuit 977 f.2d 1510 (9th Cir. 1992) [en línea] <<http://digital-law-online.info/cases/24PQ2D1561.htm>> [consulta: 07 de septiembre de 2007]

21 Article 5. Exceptions to the restricted acts.

22 Article 6. Decompilation.

23 Ver: en España, Art. 6 de Ley Nº 16/1993 sobre Propiedad Intelectual Informática de 24 de diciembre de 1993; en Francia, Artículo L122-6-1 (IV) del Código de Propiedad Intelectual; y, en Alemania, el Art. 69 de la ley de propiedad intelectual (Urheberrechtsgesetz, UrhG).

En nuestro país se le pretende incorporar al nuevo catálogo de excepciones y limitaciones al derecho de autor, en relación a los programas computacionales, en los siguientes términos:

Art. 71 P letra b). “Están permitidas, sin que se requiera remunerar al titular ni obtener su autorización [...] las actividades de ingeniería inversa sobre una copia obtenida legalmente de un programa computacional que se realicen con el único propósito de lograr la compatibilidad operativa entre programas computacionales o para fines de investigación y desarrollo”.

Como se puede apreciar, el tenor sugerido, no sólo restringe la procedencia de la descompilación para los fines de identificación de aquellos elementos que permiten alcanzar la compatibilidad operativa entre programas propietarios sino que, además, a los fines de investigación y desarrollo, gracias a lo cual la procedencia de esta ciencia en nuestro país podría hacerse extensiva a todas y cada una de las aplicaciones de la misma, analizadas con anterioridad.

7.- LA INGENIERÍA INVERSA DE SOFTWARE COMO EXCEPCIÓN

A continuación, se enunciarán una serie de principios generales en los cuales, estimamos, se sustenta todo el régimen de excepciones y limitaciones a los derechos patrimoniales de autor, esto es, aquellas que permiten la utilización de las obras por parte de la comunidad, sin resultar menester obtener la autorización previa ni remunerar a los titulares de los derechos sobre las mismas. Lo anterior cobra relevancia porque los principios que abordaremos a continuación constituyen el prisma que permite apreciar si un determinado uso o actividad realizado sobre una obra o prestación protegida -en el caso que nos interesa, la ingeniería inversa de software o descompilación- es procedente o, dicho de otra manera, reviste tal importancia que puede considerarse fundada en alguno o algunos de dichos principios, para ser elevada a la categoría de excepción o limitación de carácter legal.

a) La función social de la propiedad

La Constitución Política dispone que sólo la ley puede establecer el modo de adquirir la propiedad, de usar, gozar y disponer de ella y las limitaciones y obligaciones que deriven de su función social, la cual comprende, en lo pertinente, todo cuanto exijan los intereses generales de la Nación. El objeto sobre el cual recae el Derecho de Autor, esto es, las creaciones artísticas e intelectuales, determinan que las máximas que rigen este particular tipo de propiedad difieran considerablemente de las normas y principios generales aplicables a este derecho real en cuanto recae en otro tipo de bienes materiales e inmateriales y en ese sentido, la limitación constitucional dada por la “función social de la propiedad” basada en los intereses generales de la Nación adquiere especial relevancia desde que el acto de creación es entendido como “el proceso mediante el cual

una persona en ejercicio de su intelecto y de sus emociones, produce un objeto material o inmaterial que transmite valores, ideas y conocimiento, que contribuye, entre otras cosas, a forjar la identidad cultural de una colectividad”.²⁴

La Convención para la Protección y Promoción de la Diversidad de las Expresiones Culturales de la Organización de Naciones Unidas para la Educación, la Ciencia y la Cultura ratificada por Chile y vigente desde marzo del año en curso, reconoce “la doble dimensión económica y cultural de las actividades, bienes y servicios culturales, en tanto portadores de identidades, valores y significados [que por consiguiente] no deben tratarse como si sólo tuviesen un valor comercial”. Es en atención a esta característica de los bienes intelectuales, esto es, el constituir medios de transporte y expansión del conocimiento y la cultura dirigidos a la comunidad, que los titulares de los derechos sobre este tipo de obras deben soportar una serie de limitaciones a las facultades exclusivas de explotación derivadas de la faz patrimonial del derecho de autor para, de esta forma, posibilitar el acceso y adecuado disfrute de las mismas, por parte de los usuarios.

La ingeniería inversa de software, como limitación al derecho exclusivo de reproducción reconocido a los autores y titulares de los derechos, se encuentra justificado en los intereses generales de la nación, que, en este caso, se encuentra representada por los terceros desarrolladores de software, por los usuarios finales de este tipo de obras, conforme los beneficios del reconocimiento de esta ciencia a los que nos hemos referido a lo largo del presente estudio y por el resto de los habitantes de la república, en atención a que el desarrollo de la industria local de software producto de la innovación que el empleo de esta ciencia posibilita, lleva consigo el aumento en la demanda laboral, así como de los ingresos fiscales para su redistribución en obras públicas y de fomento.

b) Los principios recogidos de la Declaración Universal de los Derechos Humanos

Ya en la Declaración Universal de Derechos Humanos, aprobada y proclamada el 10 de diciembre de 1948, por la Asamblea General de las Naciones Unidas encontramos un reconocimiento de la necesidad de armonización de intereses entre ambos extremos de la cadena ligada a las creaciones del intelecto. De esta forma, el artículo 27 de dicha Convención consagra, por un lado, el derecho de las personas a participar de la vida cultural de la comunidad, gozar de las artes y tener acceso a los avances en las ciencias. Asimismo, se reconoce a los creadores, el derecho de autor en su faz integral, aun cuando no como un derecho absoluto.²⁵ Ambos son elevados a la categoría de De-

24 Daniel ÁLVAREZ V., “Derecho de Autor y Cultura”, [en línea] en “Los derechos de propiedad intelectual y el libre comercio”, 2005. <<http://www.derechosdigitales.org/2005/12/05/derecho-de-autor-y-cultura/>> [consulta: 29 de agosto de 2007]

25 Sofía RODRÍGUEZ, “Excepciones y limitaciones al Derecho de Autor en el Ciberespacio. Una forma de justicia alternativa para un nuevo modelo de Estado”, Universidad Extremado de Colombia, 2003, p. 48.

rechos Humanos, la de mayor entidad y trascendencia que se le puede reconocer a los sujetos de derecho.

De esta forma, el reconocimiento realizado en la Declaración Universal de Derechos Humanos, respecto al derecho de las personas a participar del progreso tecnológico y de los beneficios que de él derivan, permite sostener que la ingeniería inversa, en el apartado investigación y desarrollo, permite a terceros desarrolladores introducirse en el mercado innovando respecto a los productos existentes, beneficiando consecuentemente a los usuarios finales que gozarán de un mayor número de alternativas en ese sentido y a precios más competitivos, lo cual se torna más relevante si se considera la naturaleza de este mercado, centrado generalmente en torno a sistemas estandarizados y dominantes.

c) La regla de los tres pasos

Este principio introducido en el Convenio de Berna celebrado el 9 de septiembre de 1886, entrega a los Estados Parte una directriz a seguir al momento de introducir en sus respectivas legislaciones un marco de excepciones a los derechos de autor, propósito del derecho de reproducción de los autores.

La regla de los tres pasos constituye una guía para los Estados que se aboquen a la tarea de establecer un régimen de limitaciones a las facultades exclusivas derivadas del derecho de autor y ha sido incluida por muchas legislaciones que se han hecho cargo de esta tarea. El reconocimiento de la ingeniería inversa en gran parte del mundo como excepción o limitación a los Derechos de Autor da cuenta de la conformidad de la disciplina en estudio con este importante principio, reconocido e integrado a las diversas legislaciones de los países que concurrieron a la suscripción y ratificación del Convenio de Berna.

d) Principios extraídos del Acuerdo sobre los ADPIC para los países en desarrollo

En el marco del Acuerdo sobre los Aspectos de los Derechos de Propiedad Intelectual relacionados con el Comercio, de la OMC, del año 1994, se reconoce, en primer lugar, las necesidades especiales de los países menos adelantados Miembros, por lo que se refiere a la aplicación, a nivel nacional, de las leyes y reglamentos con la máxima flexibilidad requerida para que esos países estén en condiciones de crear una base tecnológica sólida y viable.²⁶

26 Acuerdo sobre los ADPIC. Considerandos. Montevideo. 1994. [en línea] <http://www.wto.org/spanish/docs_s/legal_s/27-trips.pdf> [consulta: 17 de octubre de 2007]

De esta forma y reconociendo la problemática antes expuesta, el Acuerdo sobre los ADPIC formula los principios, que se puede asimilar en cierta forma al abuso de derecho de autor (copyright misuse), a propósito de las limitaciones a la descompilación en las licencias de software.

Entre nuestro país, la ingeniería inversa de software es una herramienta que posibilitará la identificación, el aislamiento e incorporación de aquellos elementos que permitan la obtención de compatibilidad operativa entre dichas aplicaciones y las plataformas dominantes a nivel mundial, en pro del desarrollo de la industria local ligada a las obras de la creación, con los consecuenciales beneficios para el mercado interno (generación de empleo, mayores ingresos privados y fiscales, etc.).

La disciplina que ahora nos ocupa adquiere una relevancia para países en vías de desarrollo como Chile, en atención a que esta herramienta permite innovar sobre la base del estudio de las tecnologías importadas desde países que cuentan con niveles de desarrollo económico mucho más elevados que el nuestro. Al efecto, la ingeniería inversa sirvió como herramienta para el desarrollo de la base tecnológica de economías como la japonesa, que actualmente es una potencia en la producción y exportación de dicha clase de bienes, tangibles e intangibles.

El reconocimiento de la ingeniería inversa de software, para los fines señalados en el artículo 71 P del proyecto de reforma, propende al logro de innovación tecnológica, particularmente en el terreno de la informática: “la adquisición de nueva tecnología puede mejorar la competitividad de un país y facilitar su progreso económico [...]. [A] través de la ingeniería [inversa] es posible para un país en desarrollo imitar un producto nuevo, siempre que no resulte muy costoso ni demande demasiado capital”.²⁷

El potencial de la tecnología informática, específicamente del software, ha sido apreciado por la actual administración, al momento de asignar un tenor amplio a la ingeniería inversa sobre programas computacionales, supeditándola no sólo al logro de compatibilidad operativa, sino también a los fines de investigación y desarrollo. Uno de los objetivos, planteados por el Gobierno de Chile, por boca del Ministerio de Economía, es que la industria informática nacional, hacia el año 2010 logre consolidarse como un referente a nivel regional, una plataforma de servicios ligados a dicho mercado para, de esta forma, ir alcanzando cotas de desarrollo cada vez mayores.²⁸

27 Pablo RUIZ-TAGLE, “Propiedad Intelectual y Contratos”, Ed. Jurídica, Santiago, 2005, p. 23.

28 Gobierno de Chile Ministerio de Economía. Prospectiva Chile 2010. N°6: “La industria chilena del software”, 2000, pp. 6-11.

8.- CONCLUSIONES

El reconocimiento de la ingeniería inversa de software para fines de obtención de compatibilidad operativa constituye un paso muy importante en la puesta al día de nuestra legislación autoral. Dicha excepción se encuentra consagrada desde hace mucho tiempo en países más desarrollados: en Europa, en su normativa comunitaria; en el sistema norteamericano, por la vía jurisprudencial desde el precedente de *Sega v. Accolade*. Por ello, resulta indispensable que los desarrolladores independientes de software de nuestro país cuenten con una herramienta legal similar, que les permita ingresar en el competitivo mercado del software con un pie de igualdad y potenciados, además, con los fines de investigación y desarrollo.

La ingeniería inversa constituye una ciencia de carácter neutro, que puede ser usada para fines tanto lícitos como antijurídicos. Ella permite la traducción de un programa computacional de cualquier naturaleza (de sistema, de aplicación, etc.) desde el lenguaje de máquina propio del código objeto, a un lenguaje de alto nivel (aproximación del código fuente) entendible por los seres humanos. Habiéndose obtenido la información por estos medios los titulares de los derechos de autor no quedan desamparados por la legislación autoral: si quien practica el proceso de descompilación utiliza la información obtenida para el desarrollo de una obra derivada, o bien si desarrolla una obra substancialmente similar, dichos actos constituyen una infracción a las leyes sobre derecho de autor, en relación a la facultad exclusiva de reproducción que detentan los titulares sobre este tipo de obras. Por ello, se ha recalcado a lo largo de esta obra, que el empleo de la ingeniería inversa para la aprehensión de ideas, procedimientos, métodos de operación y conceptos generales, que no pueden estimarse protegidos en razón de haberse “fusionado” con la idea (propósito del programa) o por tratarse de elementos de uso común en la práctica comercial o por pertenecer al dominio público, es totalmente lícito desde que el monopolio garantido al titular de los derechos sobre el programa no se extiende a dichos elementos.

El reconocimiento de la ingeniería inversa de software, desde un punto de vista jurídico y económico, es recomendable por varios motivos:

- a) Esta ciencia ha servido para sustentar el desarrollo de la base tecnológica de economías que, en la actualidad, gozan de una presencia importante en los mercados mundiales, como la japonesa, con todos los beneficios que dicha participación acarrea a sus nacionales.
- b) Respecto de la descompilación para fines de obtención de compatibilidad operativa, su reconocimiento trae aparejados muchos beneficios para la comunidad representada por los usuarios finales y desarrolladores independientes, los cuales básicamente pueden resumirse en la posibilidad de los primeros de

poder optar por un mayor número de aplicaciones en el mercado, a precios más competitivos y con mejores alternativas de soporte técnico, en atención a la cercanía con los desarrolladores locales e independientes de este tipo de nuevas obras. Respecto de los segundos, se posibilita el ingreso a un mercado caracterizado por la estandarización en torno a un producto único.

- c) Respecto de la descompilación para los fines de desarrollo de programas competitivos, estimamos que ella resulta plausible en el terreno del mercado de las aplicaciones, especialmente las desarrolladas para mercados donde nuestro país tiene una presencia importante a nivel regional, como la industria minera, agrícola, telecomunicaciones y banca. El reconocimiento de esta disciplina para fines de investigación y desarrollo, es una poderosa herramienta jurídica puesta a disposición de los desarrolladores nacionales, para el fomento de la industria informática local.
- d) A nivel comparativo, la ingeniería inversa de software se encuentra ampliamente acogida en la legislación comparada. Si economías mucho más desarrolladas que la nuestra, en materia de software, gozan de una excepción en cuya virtud la descompilación llevada a cabo en el marco de un proceso de ingeniería inversa es una excepción válida o un uso legítimo o razonable de un programa computacional, parece lógico que en un mercado en desarrollo como el nuestro, especialmente en relación con las políticas de fomento a que debe propender el país, se cuente con una idéntica herramienta jurídica en beneficio de los desarrolladores de software locales.