

# Las patentes de software y sus efectos en Europa

Joaquín Seoane Pascual  
Profesor Titular de Universidad  
Departamento de Ingeniería de Sistemas Telemáticos  
Universidad Politécnica de Madrid  
jseoane@dit.upm.es

Ramón García Fernández  
Investigador  
Departamento de Química Física I  
Universidad Complutense de Madrid  
ramon@juguete.quim.ucm.es

Diciembre de 2000

# Índice General

<b>1</b>	<b>Introducción</b>	<b>3</b>
<b>2</b>	<b>El éxito de Microsoft: convertir capital en software que funciona</b>	<b>4</b>
<b>3</b>	<b>El desarrollo de software</b>	<b>5</b>
<b>4</b>	<b>El problema de la obviedad</b>	<b>6</b>
4.1	Historia del problema de obviedad . . . . .	7
4.2	El caso de la Oficina Europea de Patentes (EPO) . . . . .	8
4.3	Haciendo que una patente obvia no lo parezca . . . . .	9
4.4	Casos mixtos: el 1-Click de Amazon . . . . .	10
<b>5</b>	<b>Patentes sobre la idea general de una aplicación</b>	<b>11</b>
<b>6</b>	<b>Patentes sobre métodos no obvios</b>	<b>12</b>
<b>7</b>	<b>Patentes contra la compatibilidad</b>	<b>13</b>
<b>8</b>	<b>Interfaces de usuario</b>	<b>16</b>
<b>9</b>	<b>Patentes como amenazas</b>	<b>17</b>
9.1	Patentes defensivas . . . . .	18
<b>10</b>	<b>Conclusiones</b>	<b>19</b>
<b>11</b>	<b>Agradecimientos</b>	<b>19</b>

# 1 Introducción

Actualmente se está debatiendo en Europa si el software debe ser patentable. Hemos preparado este informe para ofrecer nuestro conocimiento y puntos de vista.

Buena parte del debate actual sobre las patentes de software (tanto a favor como en contra) está basado en argumentos teóricos, es decir, en cómo las patentes *deberían* afectar al desarrollo tecnológico. Estos argumentos no tienen en cuenta la verdadera naturaleza de la industria del software. Este informe está basado en testimonios sobre *hechos reales*, obtenidos a partir de la abundante experiencia de los Estados Unidos, donde el software es patentable desde hace mucho tiempo.

En primer lugar mostraremos que el éxito de una empresa de software no se basa tanto en concebir y llevar a la práctica ideas nuevas como en su capacidad para desarrollar productos de calidad basados en ideas ya conocidas. Para ello incluimos el testimonio de Joel Spolsky, que fue jefe de programa de Excel 5.0, y que nos da su visión de las razones del **éxito** de empresas como **Microsoft** y Oracle.

A continuación explicaremos cómo un programa complejo se construye fundamentalmente a partir de **numerosos componentes elementales**, muchos de los cuales plantea problemas que un programador competente es capaz de solucionar en poco tiempo, por lo que se pueden considerar triviales. Estas soluciones se encuentran una y otra vez para resolver problemas similares. Sin embargo, si estas soluciones fueran susceptibles de ser patentadas, el trabajo del programador se vería tremendamente dificultado por la necesidad de verificar si cada solución encontrada está patentada, negociar el derecho de uso de la patente, o soslayarla buscando una solución alternativa, posiblemente más complicada e ineficiente.

Luego analizaremos por qué se conceden muchas patentes de software para soluciones son muy **simples**, desde el punto de vista de un programador. Veremos cómo un buen abogado puede hacer que un invento obvio parezca complicado y original, y casos concretos de cómo tanto una oficina de patentes como los tribunales pueden darle la razón.

Comentaremos también otro tipo de patentes generalmente muy simples y siempre muy dañinas, que son las que se aplican a la **idea general de un programa**, ya que muchas veces impiden a otros resolver el mismo problema, y van por lo tanto en contra de la libre competencia y la evolución incremental de las características de las aplicaciones.

Aunque el problema más importante con que nos enfrentamos es la concesión de patentes sobre métodos o ideas muy simples, también plantean graves problemas cuando se conceden sobre inventos más complicados. Las más graves son aquéllas cuya finalidad es impedir la creación de **programas compatibles**. Explicaremos los tipos más frecuentes y *por qué son muy peligrosas para la industria europea del software*.

Con frecuencia, se conceden patentes sobre **interfaces de usuario**, es decir, sobre la forma en que un usuario se comunica con un programa. Hay serios problemas de competencia relacionados con la propiedad intelectual de interfaces de usuario. Por este motivo fueron excluidas de la protección de derechos de autor en Estados Unidos, España y otros países europeos. Sin embargo, se está permitiendo patentar interfaces de usuario.

Veremos también por qué las patentes son útiles como amenazas. *Un juicio de patentes es tan caro que es más rentable pagar por una patente inválida que defender el juicio (incluso ganándolo)*. Por esta razón, muchas compañías obtienen patentes con el único objetivo de defenderse de otras patentes. Estas patentes se conocen como **patentes defensivas**.

Terminaremos este informe con unas conclusiones generales sobre las consecuencias adversas de la patentabilidad del software en la industria en general y en la europea en particular.

## 2 El éxito de Microsoft: convertir capital en software que funciona

Joel Spolsky fue jefe de programa del equipo de la hoja de cálculo Microsoft Excel entre 1991 y 1994. Fue responsable de la versión 5.0, la primera versión de Excel que superó a las demás en ventas. Además desarrolló “Visual Basic para Aplicaciones”. En este ensayo explica las que según él son las razones más importantes del éxito de esta empresa y que no tienen nada que ver con las patentes<sup>1</sup>.

*Estoy convencido de que la mayoría de la gente piensa en las empresas de software al revés. La creencia común es que cuando se construye una empresa de software, el objetivo es encontrar una idea fantástica que resuelve un problema que no ha sido resuelto antes, programarla, y hacer una fortuna. Llamaremos a esto la creencia “hacer un juguete nuevo”. Pero el verdadero objetivo para una empresa de software debería ser convertir capital en software que funciona. Si usted entiende esto, es más fácil tomar las decisiones estratégicas correctas.*

*El problema con “hacer un juguete nuevo” es que no hay muchas pruebas de que funcione. Para empezar, muchas de las empresas con más éxito (Microsoft y Oracle, por ejemplo) no “innovan” realmente, en el sentido de que estén resolviendo problemas que no hayan sido solucionados antes. En cualquier mercado, es extremadamente raro que se pueda mantener una invención para uno mismo. Todo el mundo tiene competencia. Los inversores inexpertos de Wall Street y los abogados que están empezando una empresa de alta tecnología tienden a decir que se pueden proteger de esto con patentes. No. A duras penas puedo pensar en un sólo caso de una compañía que se haya protegido con éxito de sus competidores gracias a una patente (Stac es el único caso que puedo recordar, y ¿dónde demonios está?).*

*El siguiente problema de la creencia “hacer un juguete nuevo” es que hemos llegado a un estado con el software para Internet donde hay mucho dinero buscando cazar las mismas ideas torpes. Llamémoslo el fenómeno “servidores gratuitos de disco”: de repente treinta y siete compañías aparecen ofreciendo exactamente el mismo servicio gratis. Hay muchísimos ejemplos de esto. Pajarerías.com. Alquiler y envío de películas. Webs de cosméticos. Cuando esto ocurre, el reto del negocio pasa de ser un reto tecnológico, que necesita buenos programadores, a un reto de marketing, que necesita de alguna manera la capacidad de adelantar a los demás competidores y establecer una marca, algo extremadamente improbable. No sólo eso, sino que el dinero de las empresas de capital-riesgo es impaciente. Eso significa que las inversiones que necesitan mucho tiempo para desarrollarse no van a encontrar financiación, y esta es la razón por la que cualquier cosa interesante o difícil de copiar no va a ser financiada. Una razón de la existencia de más de treinta compañías cuyo objetivo es proporcionar disco duro gratis en Internet es que escribir el programa para ese servicio es muy fácil.*

*Hay una forma diferente de concebir el desarrollo de software. Imagínesse que el objetivo de su empresa de software **no** es resolver un problema específico, sino ser capaz de **convertir dinero en código mediante programadores**. Eso suena algo extraño, pero siga leyendo. Una empresa de software debe pensar en contratar al personal adecuado como su **problema número uno**. Si usted tiene éxito, esto puede resolver cualquier otro problema. Contrate personas inteligentes y producirán buenos programas que podrá vender y sacarles dinero. Todo lo demás viene a continuación. Microsoft es capaz de aniquilar a sus competidores porque es capaz de poner a trabajar a muchos programadores. Cuando Microsoft publicó Internet Explorer 3.0, justo después de Internet Explorer 2.0, fue sorprendente el buen trabajo que habían hecho. No sólo el*

---

<sup>1</sup>Este ensayo es ©Joel Spolsky, 2000. Incluido, adaptado y traducido con autorización. La versión original se puede encontrar en [http://joel.edithispage.com/stories/storyReader\\$17](http://joel.edithispage.com/stories/storyReader$17). La dirección de correo electrónico de Joel Spolsky es [spolsky@panix.com](mailto:spolsky@panix.com)

*programa hacía todo lo que hacía el navegador de Netscape, sino que añadieron algunas características nuevas, y todo ello lo hicieron con una arquitectura robusta y estratégica. Aunque es verdad que Microsoft utilizó su sistema operativo para promocionar su navegador, también es verdad que no habrían llegado muy lejos si su navegador no fuera excelente. (Un ejemplo: aunque Windows puede reproducir archivos MP3, todo el mundo que conozco utiliza WinAmp, no el “Reproductor de Medios de Windows”, para escucharlos. Aunque todo el mundo tiene “Microsoft Network” en el escritorio, todo el mundo usa AOL. Antes, cuando el navegador integrado con Windows era una basura, Netscape tenía el 80% de cuota de mercado. Así que dejen de molestar con el poder de integrar un programa en el sistema.)*

[...]

*Si quiere ser el restaurante número uno en la ciudad, tiene que preocuparse de conseguir los mejores cocineros y los mejores ingredientes. Usted es una factoría que convierte ingredientes brutos y cocineros en una cena. Si usted es una productora de películas, tiene que preocuparse de conseguir los mejores actores, directores y guionistas. Usted es una factoría que convierte talento en entretenimiento. Así que si usted es una empresa de software, usted tiene que tener los mejores cocineros y directores: usted es una empresa que convierte talento en código. El talento adecuado sabe como hacer el código adecuado que le dará éxito.*

### 3 El desarrollo de software

El proceso de desarrollo de programas consta de varias etapas. Una vez identificando un problema a resolver y la idea general de cómo hacerlo, se especifica su funcionalidad detalladamente. Seguidamente empieza la programación propiamente dicha, descomponiendo el problema computacional en muchas piezas pequeñas e interrelacionadas. Los programadores escriben el código de cada una de esas partes pequeñas, probándolas y depurando sus errores por separado. Finalmente se integran todas las partes y se prueban en conjunto. Una vez superadas las pruebas de integración, el programa se instala o distribuye, comenzando la fase de mantenimiento, en la que se corrigen errores descubiertos por los usuarios, se modifica la funcionalidad según las necesidades manifestadas por éstos y se hace evolucionar el producto. Esta fase es la más larga y costosa, y es donde más se manifiesta la calidad y competencia de un fabricante de software.

Vemos pues que programar consiste en descomponer un problema en muchas partes, cada una de las cuales se convierte en una sucesión de instrucciones de programación. Cada una de las partes tiene escaso valor práctico tomada aisladamente, y sólo la combinación de todas ellas, el programa completo, es útil para el consumidor.

La resolución del problema planteado por cada una de las partes en que se ha descompuesto un programa puede ser sencilla o complicada, fácil o difícil. Si es complicada, se vuelve a descomponer en partes más sencillas. Si es fácil se resuelve con el sentido común y los recursos que todo programador profesional debe poseer. Si es difícil, puede requerir una tarea de investigación para obtener un algoritmo apropiado <sup>2</sup>.

La inmensa mayoría de los pequeños problemas que resuelve un programador pertenecen a la categoría de **fáciles**, si el programador está suficientemente cualificado. La habilidad de encontrar una combinación de instrucciones de ordenador para conseguir un objetivo es esencial en un buen programador. He aquí una prueba: *en el concurso de programación de ACM, los participantes*

---

<sup>2</sup>Aunque la legislación actual no permite patentar *algoritmos* tal cual, en la práctica sí lo son. Como la mayoría de los algoritmos tienen una función práctica son también *métodos* y, por tanto, son patentables

*deben resolver problemas realmente difíciles en 6 horas*<sup>3</sup>. Sugerimos que comparen una patente de software, por ejemplo la patente de EE.UU. 4197590 sobre el uso de la operación lógica *o exclusivo* para dibujar cursores (también otorgada por las oficinas de patentes de Francia y Reino Unido) con un problema cualquiera de los últimos concursos mundiales de programación de la ACM<sup>4</sup>.

Además, sean fáciles o difíciles los pequeños problemas que resuelve un programador, suelen tener solamente una solución o un conjunto pequeño de soluciones razonables. Si las soluciones a estos pequeños problemas fueran patentables, la **infracción accidental** de una patente sería bastante probable. Además a un programador le resultaría mucho más sencillo resolver el problema por sí mismo que buscar en la base de datos de patentes si el problema ya está resuelto por alguna patente caducada o vigente. En el caso de estar vigente, tendría además el trabajo adicional de negociar la licencia o intentar soslayarla buscando una solución distinta.

La gran probabilidad de infracciones accidentales de patentes puede obligar a los fabricantes de software a ocultar aún más los fuentes de sus programas, ya que esto facilitaría las demandas contra ellos. Esto tiene graves consecuencias de seguridad y mantenibilidad. De seguridad porque hay aplicaciones de las que depende la vida y la salud de las personas, y los que hacen uso de ellas no pueden permitirse el lujo de no saber cómo funcionan. De mantenibilidad porque muchas veces un cliente necesita modificaciones de un programa que el que lo ha producido no puede realizar.

## 4 El problema de la obviedad

Una de las características más criticadas al sistema de patentes es que se conceden patentes sobre inventos muy simples, prácticamente triviales. Hasta aquí nos hemos referido a las que afectan a partes pequeñas y sencillas de un programa, pero también se puede decir de otras, como las patentes sobre interfaces de usuario y las patentes sobre la idea global de un programa, que analizaremos en apartados específicos de este documento. *Dar un monopolio de 20 años por un método para resolver un problema que requiere pocas horas para volverlo a encontrar o por una idea que aparece en un momento de inspiración y que se le puede ocurrir a cualquiera, no nos parece beneficioso para la sociedad.*

Los defensores de las patentes de software atribuyen el problema a la dificultad de los examinadores de encontrar si hay inventos anteriores. No estamos de acuerdo. *El problema es que la ley sólo exige que el método patentado “sea nuevo, un avance técnico y tenga carácter inventivo”*. El trabajo de unas horas necesario para resolver un problema computacional normal puede cumplir este criterio, aunque no debería ser patentable por las razones que hemos dicho más arriba. A pesar de las excusas habituales relacionadas con la dificultad de buscar precedentes de cada invento, ése no es el problema. El problema real es que *tanto la oficina de patentes de Estados Unidos como la Oficina Europea de Patentes tienen una filosofía de que cualquier método, por simple que sea, debe ser patentable.*

Dada la complejidad tecnológica de la programación, es difícil que los que no son programadores entiendan si un método es sencillo o no. *Creemos que esta es la razón por la que muchos abogados y gobernantes están a favor de las patentes de software.* He aquí una prueba: en Estados Unidos, cuando la Oficina de Patentes y Marcas comenzó a conceder patentes obvias sobre métodos de negocio, empezó a haber preocupación sobre el tema; en este momento se está discutiendo en el

---

<sup>3</sup>ACM (Association for Computer Machinery) es una organización mundial de profesionales y estudiantes de Informática. Este concurso es de ingenio y rapidez en la resolución de problemas. Consta 8 ejercicios a resolver en 6 horas. Habitualmente ningún participante termina los 8, pero no es normal hacer menos de 4. La página de Internet del concurso está en <http://acm.baylor.edu/acmicpc/>

<sup>4</sup>Disponibles en Internet en <http://acm.baylor.edu/Past/PastProblems.html>.

Congreso de ese país una ley para restringir las patentes sobre métodos de negocios. Sin embargo, esa oficina ha estado concediendo patentes obvias sobre software desde hace mucho tiempo y no ha habido casi ningún político que se haya quejado<sup>5</sup>. La razón es que cualquiera puede entender que una patente sobre un método de negocio es trivial, pero sólo un programador puede entender una patente de software.

El problema es conocido desde hace mucho tiempo (hubo una consulta pública en 1994 y el problema todavía sigue). He aquí un extracto de un artículo de la revista Red Herring<sup>6</sup> :

*John Barton, un profesor de Derecho de la Universidad de Stanford, propone una corrección. Recomienda subir por un factor de 10 el estándar de “nuevo y no obvio” necesario para una patente. Esto resultaría, dice, en menos “patentes-molestia”, y sería beneficioso para la innovación. “Reduciría el número de patentes que empresas grandes pueden usar contra las más pequeñas”, dice él. “Las empresas grandes que usan grandes carteras de patentes para obtener pagos de otras se están convirtiendo en un serio problema en la comunidad de profesionales de la Informática”.*

*(Red Herring, mayo de 1999, por Luc Hatlestad)*

El bajo nivel exigido por la Oficina de Patentes y Marcas de EEUU es un hecho muy conocido. Así, los despachos jurídicos (que suelen ser partidarios de las patentes de software) no dudan en usarlo. A continuación incluimos un extracto de las instrucciones de un bufete de abogados a sus clientes<sup>7</sup>:

*(Hemos puesto en negrita la parte que nos interesa para esta discusión)*

*[Pregunta] ¿Qué debo tener en cuenta para decidir si solicito una patente?*

*[Respuesta] Primero, considere si la invención proporciona una ventaja competitiva. Si es así, la protección de la patente puede ser necesaria para proteger su inversión. **No subestime su invención. Aunque una invención puede parecerle simple, puede que sea patentable. Debe evitar autocensurarse.** No sólo podría perder derechos valiosos, sino que un competidor podría obtener una patente sobre su invención. En ese caso, usted podría tener que gastar mucho dinero en defenderse de ella.*

El problema es tan serio que hay empresas conocidas habitualmente como **empresas de litigación**, cuya única actividad es patentar ideas muy simples para que cuando a alguien se le ocurra la misma idea exigir dinero.

## 4.1 Historia del problema de obviedad

El problema de la obviedad se planteó el siglo pasado en EE.UU. En aquellos tiempos, la oficina de patentes de ese país actuaba como ahora, permitiendo patentar cualquier idea nueva, aunque fuera muy simple. Por ejemplo, en el año 1867 la Oficina de Patentes y Marcas de USA (USPTO) concedió una patente a “un libro de registro de hotel con los márgenes de las hojas ocupados por anuncios”<sup>8</sup>. Los negocios del siglo pasado utilizaron estas patentes para dominar mercados y eliminar competidores. Un ejemplo fueron las “Grandes Guerras de Telégrafos” de los años 1870.

<sup>5</sup>Aunque sí ha habido numerosas quejas de programadores y empresas.

<sup>6</sup>El artículo completo está en <http://www.redherring.com/mag/issue66/news-sue.html>

<sup>7</sup>El despacho es Wolf, Greenfield & Sacks. Estas instrucciones están en su página de web, en <http://www.wgslaw.com/news/faq/qa-soft.html>.

<sup>8</sup>Esta patente la hemos encontrado en el testimonio del director de la USPTO durante un debate. Está en Internet en <http://www.oreillynet.com/pub/a/patents/2000/10/23/isoc.html?page=2>

Alrededor de los años 30, el gobierno de EE.UU. cambió su política y empezó a ver las patentes como un mecanismo de monopolio. Con la única excepción de la industria farmacéutica, la industria empezó a restringir el uso de las patentes a fines únicamente defensivos.

Pero a finales de este siglo se volvió a la situación anterior. En 1982 se creó un nuevo tribunal, “Court of Appeals of the Federal Circuit” (CAFC), con doce jueces defensores de permitir más patentes. Antes de la creación de CAFC, el 75% de las demandas que incluían patentes eran denegadas. Hoy cerca del 72% de las demandas de patentes son aceptadas, y las indemnizaciones por infracción son sustancialmente más altas que antes. Además, cambió la doctrina del Tribunal Supremo de EE.UU. que había visto las patentes como anticompetitivas y en cambio desde entonces ha estado promoviendo el máximo poder para la propiedad intelectual. Todo esto ha relajado los requisitos para que una patente sea válida en los tribunales.

En los últimos tiempos se está abundando más aún en esta dirección, y los tribunales están aceptando patentes que rozan el ridículo. *Queda claro que la concesión de muchas patentes de baja calidad no es culpa de la dificultad de encontrar inventos anteriores. Se trata de una decisión política.*

## 4.2 El caso de la Oficina Europea de Patentes (EPO)

La filosofía de la Oficina Europea de Patentes es similar a la de EE.UU.: toda idea original, aunque sea muy simple, debe ser patentable. Veamos, por ejemplo, la patente EPO 926584:

*Sistema para controlar el uso de un artículo de software*

*Resumen:*

*Un sistema para controlar el uso de un artículo de software, tal como un programa o datos, en una red de ordenadores, que está compuesta al menos de un servidor y de una pluralidad de ordenadores locales, donde cada uno de los ordenadores locales tiene medios para bajar un artículo de software del servidor al ordenador local. El sistema incluye mecanismos para controlar el acceso, para que el proveedor del artículo pueda tener control sobre el artículo bajado.*

*Inventos:*

*1. Sistema para controlar el uso de un artículo de software tal como un programa o datos. En una red de ordenadores que consta al menos de un servidor y de una pluralidad de ordenadores locales que tienen mecanismos para bajar un artículo del servidor al ordenador local, dicho sistema incluye mecanismos para controlar el acceso al artículo bajado.*

*[...] (El resto de apartados no son relevantes para esta discusión. Recordemos que cada apartado representa un invento que puede ser utilizado en un tribunal).*

Veamos un poco más detalladamente esta patente. En el resumen tenemos una descripción de cómo funciona un servidor en Internet: un ordenador central (que pertenece habitualmente a una empresa que ofrece un servicio) y ordenadores “locales” de cliente que se conectan para obtener ese servicio.

Lo único novedoso que hay tanto en el resumen como en el apartado 1 es que “sería útil tener algún mecanismo para poder restringir lo que el usuario hace con ese software que se baja”, pero no se nos dice ni siquiera cómo es ese mecanismo. El defecto de esta patente no es que no tenga nada nuevo, es que no tiene *nada* (en otros apartados hay mecanismos de control de acceso propuestos, pero esos apartados son “invenciones” independientes en caso de un juicio). *Se ha concedido una patente a una idea que cualquiera que hubiera tenido esa necesidad habría podido encontrar.*

La Oficina Europea de Patentes también ha aceptado la patente del 1-Click de Amazon, que discutiremos más adelante en la sección 4.4.

### 4.3 Haciendo que una patente obvia no lo parezca

Un buen abogado de patentes puede hacer que una idea obvia parezca un método complejo. Para ello, *la única forma posible de usar la idea se describe con mucho detalle y con un lenguaje técnico complicado*. Esto se explica mejor con un ejemplo: la patente de EE.UU. 5963916<sup>9</sup>.

El “método” de esta patente es que “es útil para un cliente escuchar un trozo de una canción para decidir si debe comprarla”. Aplicamos esta idea a una tienda de música por Internet. Describiendo con mucho detalle cómo funciona normalmente un servidor de web (aplicándolo para este caso particular) convertimos esta idea en un método sofisticado. Veamos la patente:

*1. Un método para permitir a un usuario remoto escuchar un fragmento de un producto musical grabado de un sitio web en red que contiene fragmentos preseleccionados de varios productos de música grabada, usando un ordenador, una pantalla de ordenador y un enlace de telecomunicaciones entre el ordenador del usuario remoto y el sitio web en red, constando este método de los pasos de:*

- a) usar el ordenador remoto del usuario para establecer un enlace de telecomunicaciones con el sitio web en red, donde el sitio web en red consta de (i) un servidor huésped central acoplado a una red de comunicaciones para recibir y transmitir el fragmento preseleccionado del producto de música grabada a petición del usuario remoto y (ii) un dispositivo de almacenamiento central para almacenar fragmentos preseleccionados de una pluralidad de varios productos de música grabada;*
- b) transmitir la identificación del usuario desde el ordenador remoto del usuario al servidor huésped central, permitiendo así al servidor huésped central identificar y seguir la pista del progreso;*
- c) elegir al menos un fragmento preseleccionado de los productos de música grabada del servidor huésped central;*
- d) recibir el fragmento elegido de los productos de música grabada; e*
- e) interactivamente escuchar el fragmento preseleccionado elegido del producto de música grabada.*

Ahora vamos a ver esta patente con detalle.

*1. Un método para permitir a un usuario remoto escuchar un fragmento de un producto musical grabado de un sitio web en red que contiene fragmentos preseleccionados de varios productos de música grabada, usando un ordenador, una pantalla de ordenador y un enlace de telecomunicaciones entre el ordenador del usuario remoto y el sitio web en red*

El cliente necesita un ordenador para estar conectado a Internet. Habitualmente los ordenadores personales incluyen una pantalla. Y tanto el vendedor de música como el cliente están en Internet, que actúa como “enlace de telecomunicaciones”.

- a) usar el ordenador remoto del usuario para establecer un enlace de telecomunicaciones con el sitio web en red,*

---

<sup>9</sup>Este ejemplo se debe a Richard Stallman.

El método estándar para usar la web hoy consiste en que un navegador en el ordenador del usuario se conecte (“establecer un enlace de telecomunicaciones”) con un servidor de web.

*donde el sitio web en red consta de (i) un servidor huésped central acoplado a una red de comunicaciones para recibir y transmitir el fragmento preseleccionado del producto de música grabada a petición del usuario remoto y (ii) un dispositivo de almacenamiento central para almacenar fragmentos preseleccionados de una pluralidad de varios productos de música grabada;*

Aquí se nos informa de que el sitio web es un ordenador (todos lo son). Como este ordenador espera peticiones de clientes es un servidor (en la versión original en inglés, “central host server” suena mucho más técnico que servidor a secas). Tiene un disco duro (“dispositivo de almacenamiento”) y una conexión a Internet. También se nos recuerda que la función del ordenador es enviar y recibir datos (las conexiones de red se usan mandando y recibiendo datos). El disco duro cumple su función habitual de almacenar datos, en este caso música.

*b) transmitir la identificación del usuario desde el ordenador remoto del usuario al servidor huésped central, permitiendo así al servidor huésped central identificar y seguir la pista del progreso;*

Esta es una técnica convencional en servidores de web, conocida como “galletas”. Las galletas son un mecanismo que sirve exactamente para identificar a los usuarios (por ejemplo, para que cada usuario puede ver una página de web con preferencias distintas). Todo lo que dice este párrafo es que “usamos galletas”.

*c) elegir al menos un fragmento preseleccionado de los productos de música grabada del servidor huésped central;*

Antes de que el usuario escuche el fragmento de muestra de la música debe elegir una canción.

*d) recibir el fragmento elegido de los productos de música grabada; e*

Cuando el usuario pulsa un enlace en una página de web, su ordenador recibe el contenido de esta página. Así funciona la web.

*e) interactivamente escuchar el fragmento preseleccionado elegido del producto de música grabada.*

Y finalmente el usuario escucha la música. Eso sucede automáticamente con muchos navegadores de web cuando el usuario pulsa un enlace a un archivo de sonido.

Con la técnica de esta patente, es posible patentar cualquier idea simple.

#### **4.4 Casos mixtos: el 1-Click de Amazon**

Hay muchos casos intermedios de obviedad de patentes. Se trata de patentes que tienen una parte obvia y otra que no lo es, pero que no es patentable. Su finalidad es saltarse las restricciones sobre patentabilidad. Por ejemplo, de esta manera se pueden patentar ideas, que en sí mismas no son patentables.

Un buen ejemplo es la famosa patente 1-Click de Amazon, aceptada por las oficinas de patentes de Estados Unidos y Europa (US 5960411, EPO 902381). La patente cubre el uso del sistema de “galletas” de Internet para que el usuario no tenga que volver a poner su dirección y tarjeta de crédito cada vez que compra. Esta idea es evidente, pues fue una de las finalidades con que se creó el sistema de “galletas” (parte del estándar HTTP, que está libre de patentes) y así lo reconoce el presidente de Amazon, como veremos a continuación.

Sin embargo, hay algo de original en el sistema 1-Click. El interfaz de usuario de compra de libros en Amazon permite pedir un libro con una sola pulsación de un botón. Hasta ese momento, las tiendas por Internet usaban la metáfora del carrito de compra. Esto podemos confirmarlo leyendo lo que dijo el presidente de esta compañía<sup>10</sup>:

*Antes de nada, Jeff quiso explicar por qué creía que 1-Click era lo suficientemente original como para patentarlo. No tiene nada que ver con el método informático –él admite que es bastante trivial duplicarlo–, sino con la formulación del problema. Cuando él propuso el sistema 1-Click, todo el mundo estaba usando la metáfora del carrito de compra, porque eso es lo que se usa en el mundo real. Se coge un artículo y se lleva a la caja para comprarlo. Se dió cuenta de que en la web era posible algo muy distinto: todo lo que tiene que hacer es apuntar al artículo, y ya es suyo.*

Es verdad que Amazon ha sido la primera librería en Internet que se ha preocupado por hacer su servicio fácil de usar. Pero *aplicar a Internet técnicas de diseño de interfaces de usuario que existen desde los años 80 para los programas de ordenador no les hace inventores*<sup>11</sup>.

## 5 Patentes sobre la idea general de una aplicación

Estas patentes prohíben el desarrollo de un programa que tenga un objetivo concreto. Los defensores de esta clase de patentes dicen que el que tiene una idea de un programa debe obtener una recompensa por ella. Pero *la idea de una aplicación suele venir de observar a los usuarios, o de recibir sus peticiones*. Por ello nuestra opinión es que estas patentes no tienen suficiente trabajo detrás para ser merecidas.

Un ejemplo típico (y divertido) es la patente de EE.UU. 6049811, que cubre la idea de una aplicación para solicitar patentes. La idea procedió de observar a los posibles usuarios, como podemos deducir de este extracto de un artículo del New York Times<sup>12</sup>:

*[...] muchos inventores que envían sus solicitudes acaban volviendo a un abogado de patentes o agente después de que sus documentos hayan sido rechazados como inadecuados. Eso requiere que un inventor explique su invento al abogado para que la solicitud sea legalmente tan eficaz como sea posible. Y los conocimientos del abogado cuestan dinero.*

*James D. Petruzzi dice que los inventores pueden evitarse algunas de estas molestias y gastos con su programa. El propio Petruzzi es un abogado de patentes, pero no cree que su invento vaya a dañar a los que se dedican a eso.*

---

<sup>10</sup>El texto procede de Tim O'Reilly, que tuvo una conversación telefónica con Jeff Bezos a raíz de la polémica patente de Amazon. El texto de Tim está en [http://www.oreilly.com/ask\\_tim/bezos\\_0300.html](http://www.oreilly.com/ask_tim/bezos_0300.html)

<sup>11</sup>Por ejemplo, la primera edición del libro “Designing the User Interface” de B. Shneiderman, cuya primera edición fue publicada en 1987, explica cómo diseñar el interfaz de usuario de una aplicación a partir de la funcionalidad. Usando el método de este libro, el 1-Click es evidente.

<sup>12</sup>El artículo está disponible en Internet en <http://partners.nytimes.com/library/tech/00/05/biztech/articles/01pate.html> (para acceder es necesario inscribirse gratuitamente)

*“Durante nuestro ejercicio de abogados encontramos que hay mucha gente que no tiene los recursos necesarios para pagar un abogado de patentes para todo el proceso, declaró Petruzzi. “Este programa permite a mucha gente meterse en el sistema y solicitar patentes”.*

*(New York Times, “Product Designed to Ease Patent Process Wins Patent”, por Sabra Chartrand, 1 de mayo del 2000)*

Del último párrafo queda bastante claro que la idea procedió de la observación de sus clientes cuando ejercía de abogado.

Generalmente las características de los programas **evolucionan** según un proceso **incremental**. A medida que los usuarios adquieren experiencia con un programa, se dan cuenta de que echan de menos ciertas características. Los vendedores toman nota de ellas y las añaden a los programas. Si fuera posible patentar la idea general de un programa, esta evolución sólo sería posible si el poseedor de la patente quisiera, descartando toda posibilidad de competencia.

Casos típicos de evolución de programas los tenemos en procesadores de textos (WordStar, luego WordPerfect, y luego Microsoft Word) o en hojas de cálculo (Visicalc, luego Lotus 123, luego Excel). Por ejemplo, *Visicalc* fue la primera hoja de cálculo moderna. Según sus autores, estaba basada en programas anteriores de tratamiento de datos de negocios por columnas. El programa *123*, de la empresa Lotus, fue la siguiente hoja de cálculo. Lotus observó que los usuarios perdían mucho tiempo repitiendo largas secuencias de operaciones. Así que añadieron *macros*, un método para poder repetir una secuencia de operaciones cómodamente. Más tarde Microsoft observó que los usuarios estaban utilizando las hojas de cálculo para almacenar listas de datos (hasta entonces se creía que el uso principal era para finanzas). Así que añadieron características tales como ordenar más fácilmente datos o entrada automática. Microsoft también añadió “Visual Basic para Aplicaciones” como mecanismo alternativo a las macros. Más tarde Lotus añadió todas estas características a su hoja de cálculo.

Esto nos lleva a otro argumento: la **evolución convergente**. Si dos aplicaciones van dirigidas a solucionar el mismo problema, tendrán que acabar teniendo las mismas características. Aunque la imitación no parezca buena, en realidad es lógico. Por eso, si miramos las últimas versiones de hojas de cálculo o procesadores de texto, veremos que tienen funciones muy similares, aunque los menús o los iconos sean distintos. Si fuera posible patentar las mejoras, la evolución convergente sería imposible.

## 6 Patentes sobre métodos no obvios

No sólo las patentes sobre ideas o métodos muy simples son peligrosas. Hoy día hay millones de programadores trabajando para resolver problemas no triviales. Por ello es muy grande la probabilidad de descubrir soluciones a problemas no obvios en varios lugares casi simultáneamente. Si bien en algunos casos la inversión para llegar a esas soluciones puede ser importante, suele ser mucho más pequeña que en otras industrias, y su patentabilidad sólo recompensa al que tenga la suerte de registrarla primero, con el agravante de que la inversión para poner a producir una idea software es prácticamente nula.

Como ejemplo de invenciones simultáneas podemos citar los algoritmos de compresión inventados por Ross Williams y publicados en Abril de 1991, pero que no pudieron beneficiar a la sociedad por culpa de la patente de EE.UU. 5049881, solicitada unos meses antes y concedida después. En este caso la patente no ha contribuido a que el método fuera publicado, ya que se hizo de todos modos.

La patentabilidad del software perjudica especialmene al llamado *software libre*, ya que se vería obligado a buscar soluciones no patentadas para todas y cada una de sus funciones, dada su in-

capacidad de recolectar dinero para pagar las licencias, aunque fueran muy baratas, y su absoluta incapacidad para ocultar su uso, ya que el código es abierto.

A continuación describiremos dos tipos de patentes que pueden ser no obvias y que tienden a favorecer a los monopolios, ya que impiden hacer programas compatibles.

## 7 Patentes contra la compatibilidad

Llamamos así a patentes diseñadas para hacer imposible crear un programa compatible con otro, porque cualquier programa compatible infringe esa patente. Estas patentes son utilizadas habitualmente por empresas dominantes para atar a sus consumidores. Habitualmente las empresas europeas de software no son dominantes, por lo que estas patentes son especialmente perjudiciales para Europa.

Generalmente, estas patentes afectan a un “lenguaje” usado por un programa de ordenador para comunicarse con otro programa, o para almacenar datos que pueden ser leídos por otro programa. Muchas veces no hay una ventaja especial en usar un lenguaje sobre otro. Pero sobre todo, estas patentes *no son beneficiosas para la sociedad* porque hacen imposible la competencia.

Creemos que estas patentes deberían estar prohibidas por esta razón y por coherencia con el espíritu de la directiva comunitaria 91/250/EEC del 14 de mayo de 1991 sobre derechos de autor de programas informáticos. Esta directiva autoriza la ingeniería inversa (normalmente prohibida) de un programa cuando es necesaria para fines de compatibilidad. Sería contradictorio permitir restricciones a la compatibilidad con patentes cuando las leyes de derechos de autor están diseñadas para evitar esas restricciones. Además, la experiencia de permitir la ingeniería inversa con fines de compatibilidad ha sido positiva. Mediante ingeniería inversa<sup>13</sup> se desarrolló un servidor de archivos compatible con Microsoft Windows NT Server, llamado Samba. Este programa era más rápido que Windows NT Server, lo que obligó a Microsoft a mejorar su velocidad. La versión 4.0 service pack 3 consiguió un incremento notable de la velocidad, que Microsoft utilizó en su publicidad. Por tanto, la ingeniería inversa ha traído competencia que ha beneficiado a los usuarios.

Estas patentes se pueden clasificar según el tipo de compatibilidad que pretender impedir:

- **Formatos de archivo.** Estas patentes cubren el formato usado por una aplicación para grabar y leer datos de archivos. Por ejemplo, los procesadores de textos graban documentos en archivos y los usuarios intercambian estos archivos. Si el fabricante del procesador de textos más popular patentara el formato de sus archivos, entonces todos los usuarios se verían obligados a comprar ese procesador de textos para poder intercambiar documentos con otros usuarios. Si además ese procesador de textos sólo funcionara en un sistema operativo de la misma compañía, los usuarios también se verían obligados a usar ese sistema operativo.

Por ejemplo, el formato ASF de Microsoft, usado para sonido y vídeo, está patentado (patente de Estados Unidos 6041345). ASF se usa no sólo para guardar vídeo y sonido en un disco duro, sino para transmisión por Internet (por ello, esta patente pertenece también al grupo de protocolos de red). Gracias a esta patente no es posible desarrollar programas capaces de leer archivos o transmisiones ASF fuera de la plataforma Windows (en este sistema operativo, Microsoft proporciona bibliotecas para acceder al contenido de los archivos ASF). Esto impide adaptar aplicaciones escritas para Windows a otros sistemas operativos.

---

<sup>13</sup>Lo necesario para la compatibilidad es el protocolo de red. La mayor parte de este protocolo es público y no es necesario hacer ingeniería inversa. Pero una parte importante, los “dominios NT” ha sido mantenida en secreto por Microsoft para obligar a usuarios con redes con puestos Windows 95, 98 ó NT a que usen como servidor Windows NT Server.

Como prueba de que ésta es realmente la estrategia de Microsoft, incluimos un fragmento de la sentencia del juicio antimonopolio contra esta compañía. En él, el juez reconoce el deseo de Microsoft de que las aplicaciones multimedia estén atadas a su sistema operativo, y por esta razón atacó a RealNetworks, que ofrece servicios alternativos, independientes del sistema operativo, para aplicaciones multimedia<sup>14</sup>:

*Párrafo de la sentencia del tribunal (hemos puesto en negrita la parte relevante para esta discusión)*

*111. RealNetworks es el líder, por número de usuarios, en software para la “transmisión” de sonido y vídeo desde la Web. El software de transmisión de RealNetworks presenta un conjunto de APIs que compiten por la atención del programador con los APIs expuestos por las tecnologías de transmisión de DirectX de Microsoft. Como Apple, RealNetworks ha desarrollado versiones de su software para varios sistemas operativos. En 1997, varios ejecutivos superiores de Microsoft vieron el software de transmisión de RealNetworks con el mismo recelo con el que veían el software para ver vídeo de Apple - **como tecnología competitiva que podría convertirse en una capa intermedia, que podría, a su vez, convertirse en lo suficientemente amplia y extendida como para poder debilitar las barreras de entrada de las aplicaciones***

- **Protocolos de red.** Un protocolo de red es un lenguaje que se usa para la comunicación entre dos programas a través de una red. La finalidad de estas patentes es habitualmente obligar a los usuarios del programa dominante a comprar el otro programa a la misma empresa.

Un ejemplo curioso de protocolo de red patentado es WAP (patente de Estados Unidos 5809415), que se utiliza para el acceso a Internet desde teléfonos móviles. En este caso el propietario de la patente mantuvo en secreto su posesión mientras convenció a los fabricantes de teléfonos móviles de que se adoptara como estándar. Ahora los fabricantes de teléfonos móviles se ven obligados a pagar cantidades de dinero arbitrarias por un protocolo cuyo único valor es que todo el mundo lo usa.

- **Códigos de instrucciones de procesador.** La finalidad de estas patentes es atar las aplicaciones desarrolladas para un procesador a éste, impidiendo que otros procesadores puedan ejecutar estas aplicaciones.

El procesador es la pieza central de un ordenador, que ejecuta las aplicaciones. Cuando un programa se ejecuta en un ordenador, lo que ocurre en realidad es que el procesador va leyendo e interpretando las instrucciones de esta aplicación. La aplicación tiene órdenes que son ejecutadas por el procesador. Llamamos “código de instrucción” a cada una de esas órdenes. Para que un procesador pueda ejecutar aplicaciones escritas para otros procesadores, es necesario que sea capaz de entender todos los códigos de instrucción de estos procesadores.

*Para que un procesador tenga éxito, es necesario que haya aplicaciones que se puedan ejecutar en él.* Por ejemplo, los ordenadores Apple, a pesar de ser fáciles de usar, tienen unas ventas escasas comparados con los ordenadores compatibles PC, porque hay menos aplicaciones para un ordenador Apple que para un PC.<sup>15</sup> Intel está patentando las instrucciones del nuevo pro-

---

<sup>14</sup>En Estados Unidos hay dos sentencias. En la primera, el juez decide cuáles son los hechos probados (llamada “Finding of Facts”), y en la segunda decide la condena. Este fragmento (y los siguientes en este documento) pertenece a la primera. El documento completo se puede ver en Internet en <http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm>

<sup>15</sup>Una descripción excelente de la importancia de la compatibilidad entre procesadores y sistemas operativos está en la dirección de Internet [http://joel.edithispage.com/stories/storyReader\\$117](http://joel.edithispage.com/stories/storyReader$117).

cesador IA-64. De esta manera se asegura contra la fabricación de procesadores compatibles. Aquí mostramos un extracto del comentario de la revista EETimes<sup>16</sup>:

*[...] Intel está intentando patentar las funciones ejecutadas por instrucciones específicas. Al hacer esto, la empresa parece que está, de hecho, intentando patentar el conjunto de instrucciones IA-64.*

*[...] “No creo que ellos [Intel] vayan a licenciar estas patentes”, declaró Rich Belgard, un consultor sobre microprocesadores en Saratoga, California. “Creo que quieren proteger el IA-64 de la clonación”*

Observación: aquí *clonación* significa desarrollar un procesador compatible. Esto es distinto del significado de clonación en otras partes de este documento.

- **Interfaces para programas de aplicación (en inglés APIs).** Conforman el lenguaje usado por las aplicaciones para utilizar servicios que no realizan por sí mismas. Una aplicación utiliza el procesador para hacer cálculos, pero otro tipo de acciones, como abrir un archivo, crear una ventana o dibujar en pantalla, se hacen pidiéndoselo al sistema operativo o a otros módulos independientes. Es para esto para lo que se utilizan las APIs.

La **finalidad** de estas patentes es exactamente la misma que la de las patentes sobre códigos de instrucción de procesadores: *atar las aplicaciones a un sistema operativo*. Un sistema operativo sólo es útil si hay aplicaciones que funcionen con él<sup>15</sup>. Por ello un nuevo sistema operativo debería poder ejecutar aplicaciones de los sistemas operativos existentes, y para ello debe soportar sus interfaces. Por ejemplo, la versión 1.0 de Microsoft MS-DOS implementó las interfaces de CP/M, el sistema operativo dominante en aquella época. Esto le permitió que WordStar, uno de los programas más populares del momento, funcionara en MS-DOS. Y MS-DOS tuvo mucho éxito. Por el contrario, el moderno sistema operativo BeOS, aunque desde un punto de vista técnico excelente e *innovador*, no ha tenido éxito porque no es compatible con Windows 98, y por tanto no hay muchas aplicaciones para él.

Veamos un ejemplo. Microsoft posee la patente de Estados Unidos 6101510 sobre la idea de usar un navegador como “control”<sup>17</sup>. Se trata de una patente bastante obvia (es la combinación de dos conceptos ya existentes: un control y un navegador). Su objetivo es que las aplicaciones que utilizan el control ActiveX de Internet Explorer queden atadas al sistema operativo Windows. El deseo de Microsoft de no tener APIs para Internet independientes del sistema operativo fue la razón más importante para promocionar Internet Explorer, así como de las prácticas anticompetitivas que la llevarían a un juicio antimonopolio. Aquí mostramos un extracto de la sentencia:

*[Bill Gates] avisó a sus compañeros dentro de Microsoft que Netscape estaba “persiguiendo una estrategia multiplataforma en que mueven el API clave hacia el cliente para que el sistema operativo que hay por debajo sea reemplazable.”*<sup>18</sup>

---

<sup>16</sup>El artículo original completo está en la dirección de Internet [http://www.eet.com/story/industry/systems\\_and\\_software\\_news/OEG20001027S0028](http://www.eet.com/story/industry/systems_and_software_news/OEG20001027S0028).

<sup>17</sup>Un control es un elemento de interfaz de usuario, que se incluye en las ventanas de aplicaciones. Se caracteriza porque las instrucciones para incluirlo son similares para todos los controles de un mismo sistema de ventanas. Los sistemas de ventanas tienen una serie de controles estándar que las aplicaciones pueden incluir, tales como botones, cuadros de diálogo, ... para que todos los programas tengan un aspecto consistente.

<sup>18</sup>La idea es que con HTML, javascript y plugins se pueden hacer programas que no dependen para nada del sistema operativo.

## 8 Interfaces de usuario

La interfaz de usuario es la forma en que el usuario dialoga con un programa. Está compuesta por una serie de elementos de diálogo (menús, botones, ventanas, ...) que se disponen de una forma determinada y se utilizan con una lógica consistente.

*Las restricciones en la copia de las interfaces de usuario ata a los usuarios al producto que conocen.* Aprender a manejar un programa es una tarea costosa y difícil, sobre todo para los usuarios de más de 40 años. Hay muchas pruebas. Basta con acercarse a cualquier librería de Informática y ver la gran cantidad de libros para aprender a manejar programas. O los cursos pagados por empresas. Por ello, muchos usuarios procuran evitar cambiar de un programa a otro con un aspecto externo distinto. Este es un ejemplo de cómo un buen principio, como la protección contra copia para fomentar la innovación, es perjudicial si se aplica ciegamente.

Un buen ejemplo de esta idea es el juicio “Lotus contra Borland” en Estados Unidos. Lotus demandó a Borland porque el programa Borland Quattro Pro ofrecía la opción de usar la misma estructura de menús que Lotus 123. La razón por la que Borland copió la estructura de menús de Lotus no fue ni por falta de talento ni para ahorrar trabajo, puesto que Quattro Pro incluía también una estructura de menús original. Está bastante claro que la única razón para ofrecer la estructura de menús de Lotus 123 fue para ayudar a los usuarios que tenían experiencia con ese programa. Por esta razón, Borland ganó el juicio en el Tribunal Supremo de EE.UU. Por la misma razón, varios países (entre ellos España) tienen leyes que excluyen a las interfaces de usuario de los derechos de autor.

Aunque los interfaces de usuario no estén protegidos por leyes de derechos de autor sí se pueden patentar, y esta es una de las razones usadas a favor de las patentes de software. Esto demuestra el punto de vista, frecuente en medios jurídicos, de defender la máxima protección sin reflexionar en el efecto sobre la competencia:

*¿Por qué tantos abogados y clientes son reacios a darse cuenta o a reaccionar a este importante desarrollo en la propiedad intelectual? Una razón es la enorme diferencia en el precio entre preparar solicitudes de derechos de autor y patentes; gastar más dinero en costes jurídicos es siempre un asunto difícil, incluso si las consecuencias de no hacerlo son posiblemente mucho más caras.*

*Pero con decisiones como Computer Associates contra Altai<sup>19</sup> o Lotus contra Borland los tribunales han demostrado claramente que las leyes de derechos de autor proporcionan poca protección al software.*

*(National Law Journal, lunes 20 de abril de 1998)<sup>20</sup>*

Nuestra opinión es que las lecciones aprendidas de los derechos de autor de las interfaces de usuario se deben aplicar a las patentes. *Las patentes sobre interfaces de usuario atan a los usuarios a los programas que conocen, es decir, a las empresas dominantes (que normalmente no son europeas).*

El caso más extremo de copia de una interfaz de usuario es la **clonación**. Se llama clon de un programa a uno con la misma funcionalidad y el mismo interfaz de usuario, pero con distinto código. La clonación de software se ha usado como argumento para defender la patentabilidad del software.

---

<sup>19</sup>En el caso Computer Associates contra Altai, Altai utilizó inconscientemente un código robado de Computer Associates. Tan pronto como lo supo, Altai destruyó el código robado y reescribió el programa correspondiente. El Tribunal consideró que el comportamiento de Altai fue correcto.

<sup>20</sup>El artículo completo está en Internet en <http://www.ljx.com/practice/computer/0420softpat.html>

*La clonación pura es bastante rara. Crea la percepción de un programa copiado y daña la imagen pública de la empresa de software. No conocemos ningún caso de clonación entre las principales empresas de software (Microsoft, Borland, Corel, Lotus). Sin embargo, como hemos expuesto antes, con frecuencia varios programas que van dirigidos a solucionar el mismo problema acaban teniendo las mismas capacidades.*

## 9 Patentes como amenazas

Muchas veces las patentes se utilizan como amenazas. Los juicios de patentes son tan caros que suele ser más barato pagar por una patente que el demandado no considera válida, que demostrar la invalidez de esa patente en un juicio.

La defensa de un juicio de patentes de software en EE.UU. suele costar alrededor de medio millón de dólares (unos 95.000.000 de pesetas) (no podemos tener datos en Europa debido a la escasez de patentes de software concedidas). Este artículo de la revista Wired<sup>21</sup> explica muy bien estos problemas:

[...]

*De acuerdo con el profesor de la Universidad de Stanford John Barton, los juicios de infracción de patentes están entre la clase de juicios más caros en EE.UU. hoy, con un coste medio de un juicio de \$500.000 por invento. No es sorprendente que el coste de un seguro para proteger a las empresas contra demandas de infracción de patentes sea igualmente caro: \$50.000 (9.500.000 pesetas) por cada producto, con una deducción de \$50.000 en el caso de software multimedia, según Rob Lippincott, presidente de la Asociación de Multimedia Interactiva. “Este tipo de números es básicamente intolerable”, dice Lippincott [...]*

*“Patentemente absurdo, parte I”, revista Wired, julio de 1994, escrito por Simson Garfinkel*

El alto coste de los juicios puede arruinar a una empresa pequeña, incluso si tiene razón. He aquí otra cita de este mismo artículo de Wired:

*Una persona que se vio en el lado malo de esas infracciones fue Vern Blanchard, un programador cuya empresa fue destruida por una patente que no era válida.*

*Blanchard es el presidente de American Multi-Systems, una empresa con sede en San Diego que escribió un programa para que los jugadores profesionales de bingo pudieran jugar docenas de cartas de bingo al mismo tiempo. El programa funciona en un PC. Blanchard estaba listo para comenzar a vender su sistema, junto con tablas hechas a medida y ordenadores personales, a grandes locales de bingo, cuando uno de sus competidores demandó a American Multi-Systems por infracción de patente.*

*El caso debería haber sido archivado por dos razones, dice Blanchard. Para empezar, dice, la patente de su competidor “cubría un dispositivo como una calculadora manual”, y no un ordenador de propósito general que ejecuta un programa. Y la patente no podría incluir un ordenador de propósito general, dice, porque los programas para jugar al bingo los escriben estudiantes en cursos de introducción a la Informática. Simplemente no había nada nuevo en la técnica descrita por la patente, y la novedad es un requisito básico para la patentabilidad.*

*Sin embargo, dice Blanchard, la compañía poseedora de la patente fue capaz de convencer al juez para que concediera una resolución preliminar que expulsó del mercado al producto de American Multi-Systems.*

---

<sup>21</sup>El artículo completo está en [http://www.wired.com/wired/archive/2.07/patents.html?topic=&topic\\_set=](http://www.wired.com/wired/archive/2.07/patents.html?topic=&topic_set=)

*Eventualmente Blanchard descubrió una pieza crítica de “arte anterior” - una prueba concreta de que el invento descrito por la patente había sido pensado antes por otro – y fue capaz de convencer al juez de suprimir la resolución preliminar. Sin embargo, para entonces American Multi-Systems ya estaba arruinada con los costes de la defensa.*

*“Los jueces no son particularmente expertos en problemas técnicos”, dice Blanchard. “Cuando ven una patente suponen que es válida, como es su deber. Si la Oficina de Patentes dice que una patente es válida, por supuesto que para ellos es una patente válida”*

Las demandas de patentes son un mecanismo utilizado por empresas grandes contra empresas pequeñas<sup>6</sup>:

*Entre los ejecutivos y abogados con los que hablamos, la única descripción sincera de cómo cualquier empresa grande usa el proceso judicial para su beneficio, fue la ofrecida por Johnson. Sobre el tema de Lucent Technologies, cuyo estilo agresivo “Costa Este” se está volviendo célebre, dice: “Son increíblemente agresivos en el uso de su cartera de patentes de Bell Labs como fuente de beneficios. Están considerados en el valle [Silicon Valley] como extorsionistas... El enfoque básico de Lucent consiste en decir, ‘Somos Bell Labs, y tenemos cientos de miles de patentes, y estamos seguros de que alguna de ellas tiene que afectarte y tienes que haberla infringido. Te daremos una licencia general para todas nuestras patentes durante un período limitado de tiempo por una gran cantidad de dinero, y si no pagas, te demandaremos hasta que las vacas vengan a casa y no vuelvas a ver a tus hijos.’ No hay ningún interés en ninguna clase de relación estratégica. Están ahí para exprimerte tanto dinero como puedan”. Los portavoces de Lucent no han querido hacer ningún comentario.*

*Red Herring, mayo de 1999, artículo de Luc Hatlestad*

## 9.1 Patentes defensivas

Como el sistema de patentes permite patentes muy generales, muchas empresas usan las patentes sobre técnicas simples para que sea difícil estar en el mismo mercado que ellas. Otras empresas, para defenderse en caso de ser demandadas, hacen lo mismo: obtienen patentes sobre técnicas simples. De esta forma, en caso de ser demandadas, es muy probable que el demandante esté infringiendo sus patentes y demandan al demandante. Como los juicios son muy caros, las dos partes salen beneficiadas **licenciando recíprocamente** sus patentes. Las patentes usadas con esta finalidad se llaman **patentes defensivas**. Esto lo explica muy bien Oracle en el documento que envió a la Oficina de Patentes cuando ésta solicitó opiniones sobre la patentabilidad del software en enero de 1994<sup>22</sup>:

### *Corporación Oracle - Política de Patentes*

*Declaración de política oficial enviada por Corporación Oracle*

*Oracle se opone a la patentabilidad del software. La Empresa cree que las leyes de Derechos de Autor y las protecciones de secretos comerciales, en comparación con la ley de patentes, funcionan mejor para la protección de los desarrollos de programas de ordenador.*

*[...]*

*Desafortunadamente, como estrategia defensiva, Oracle se ha visto obligada a protegerse solicitando selectivamente aquellas patentes que presenten las mejores oportunidades de licencia recíproca entre Oracle y otras empresas que puedan alegar infracción de patentes.*

---

<sup>22</sup>El texto completo está en Internet en <http://lpf.ai.mit.edu/Patents/testimony/statements/oracle.statement.html>

[...]

*Oracle solicitó su primera patente en noviembre de 1991, no porque sintiera repentinamente que valía la pena la protección de patentes de su software; hizo esa solicitud por la preocupación de que otros inventores pudieran obtener patentes gracias a un sistema de patentes defecioso, y se pudieran encontrar en situación de debilitar la competitividad de la Empresa alegando infracción de patentes. Incluso si Oracle hubiera desarrollado un invento determinado antes y pudiera producir “arte anterior” apropiado para demostrar su razón, tendría que gastar miles de dólares en minutas de abogados y otros gastos para defender su tecnología legítimamente poseída. Por ello Oracle piensa que debe tener una cartera de patentes con la que responder a posibles agresores, para llegar a acuerdos de licenciación recíproca que eviten juicios. Oracle está obligada a destinar una parte significativa de sus recursos financieros a la protección de patentes de su propiedad, en lugar de usar estos recursos para innovar y desarrollar más sus productos de programas para ordenadores.*

## 10 Conclusiones

Las patentes de software son en general perjudiciales para la industria del software y en particular para la europea, que actualmente es muy débil, constituida por empresas pequeñas y medianas. Recapitulemos algunas razones:

- Las patentes de software tienden a reducir la competencia, en lugar de promoverla, beneficiando a las grandes empresas (no europeas) con grandes carteras de patentes y con equipos jurídicos especializados.
- A menos que se tomen medidas contra las patentes obvias, la Unión Europea se verá obligada a aceptar multitud de patentes triviales solicitadas por grandes empresas no europeas y que se convertirán en una permanente amenaza para las pequeñas y medianas empresas europeas, por la elevada probabilidad de infracción accidental y por la enorme dificultad de programar buscando patentes para cada problema a resolver.
- La industria de software europea se podría ver impedida a producir programas compatibles con los que vienen de fuera, bloqueando en muchos casos toda posibilidad de interoperar con ellos con software propio.
- También se vería obstaculizado el desarrollo de aplicaciones con funcionalidades similares a otras, dificultando o impidiendo la posibilidad de competencia. Esto es especialmente grave en el caso del comercio electrónico en Europa.
- El éxito de una empresa de software se basa en la **utilidad y calidad** de sus programas, no en la **genialidad** de sus ideas. Por ello los esfuerzos de la Unión Europea deben para proteger a su industria de software deben ir encaminados a la creación de buenos profesionales y empresas, promocionando tecnologías abiertas y obstaculizando la adopción de aquellas que generen dependencia exterior.

## 11 Agradecimientos

Queremos expresar nuestro agradecimiento a Joel Spolsky, por habernos autorizado generosamente a incluir un ensayo suyo en este informe, a Roberto Lumbreras, por habernos dado ideas interesantes,

y a Eduardo Antuña, Emilio Albesa, Ángel Álvarez, Guillermo Díez, Jose Manuel García y Susana Moreno que han revisado este documento y han contribuido a hacerlo más claro.