Cooperación sin mando: una introducción al software libre

Miquel Vidal
miquel@sindominio.net

Resumen

En las siguientes líneas explicaré algunos de los rasgos del movimiento del software libre, su modelo de desarrollo y el alcance político, ético y económico de su apuesta. Trataré de hacer una breve genealogía del movimiento, destacando sus rasgos singulares y lo que puede haber más allá del mismo, proponiendo algunas líneas de debate y señalando algunos de sus interrogantes.

Si bien el software libre no es un fenómeno nuevo ya que existe desde los orígenes de la informática, sí es relativamente reciente su modelo cooperativo de producción en red —el llamado *modelo bazar*— y el movimiento social que lo avala —la comunidad del software libre—. No ha sido hasta los últimos cinco años en que, ligado a la extensión de Internet y a la popularización de los ordenadores personales, el movimiento del software libre ha alcanzado su masa crítica, ha dejado de ser sólo cosa de algunos programadores y se ha convertido en un fenómeno de cooperación social liberada. En la época de la subsunción real de la totalidad de las fuerzas productivas bajo el capital, en la cual todo acaba valorizado en términos mercantiles, las empresas han tardado en advertirlo pero finalmente se han lanzado a la caza y captura de esta increíble máquina productiva, tal vez la mayor empresa colectiva que existe hoy día. ¿Qué es pues el software libre, que tanto interés está empezando a despertar?

1 ¿Qué es el software?

El software es una producción inmaterial del cerebro humano y tal vez una de las estructuras más complicadas que la humanidad conoce. De hecho, los expertos en computación aún no entienden del todo cómo funciona, su comportamiento, sus paradojas y sus límites.[1] Básicamente, el software es un plan de funcionamiento para un tipo especial de máquina, una máquina "virtual" o "abstracta". Una vez escrito mediante algún lenguaje de programación, el software se hace funcionar en ordenadores, que temporalmente *se convierten* en esa máquina para la que el programa sirve de plan. El software permite poner en relación al ser humano y a la máquina y también a las máquinas entre sí. Sin ese conjunto de instrucciones programadas, los ordenadores serían objetos inertes, como cajas de zapatos, sin capacidad siquiera para mostrar algo en la pantalla.

[1] El empeño por crear procesos computacionales cada vez más mecanizados y "autoconstructivos" nos ofrece algunos ejemplos de esos límites, que son los de la lógica. Gracias al genial matemático Kurt Gödel, se conocen bien algunos de esos límites en la noción de recursión, parte fundamental de la informática moderna. Uno de ellos es la irresolubilidad del "problema de la detención", que consiste en decidir, dado un ordenador arbitrario provisto de un programa y de unos datos arbitrarios, si llegará a detenerse o si quedará atrapado en un bucle infinito. Otro es la demostración de que ningún programa que no altere el sistema operativo de un ordenador será capaz de detectar todos los programas que sí lo hagan (por ejemplo, los virus).

Los ordenadores sólo procesan lenguaje binario,[2] pero para las personas este no es un modo válido de comunicarse (salvo a nivel sináptico :-). Si bien en los tiempos heroicos de los primeros ordenadores no les quedaba otro remedio que hacerlo, los programadores hace mucho que no escriben su código en lenguaje binario (denominado técnicamente "código-máquina"), pues es terriblemente tedioso, improductivo y muy sujeto a errores. Hace tiempo que los programadores escriben las instrucciones que ha de ejecutar el procesador de la máquina mediante lenguajes formales, llamados "de alto nivel", bastante cercanos al inglés, si bien con rígidas reglas sintácticas que lo asemejan a los lenguajes lógico-formales. Esto facilita enormemente la tarea de escribir programas pero, para que esas instrucciones sean comprensibles para el procesador, deben ser convertidas antes a código-máquina. Esa *conversión* se realiza cómodamente con programas especiales, llamados compiladores. A lo que escribe el programador se le denomina "código-fuente". Al resultado de la "conversión" (compilación) en lenguaje-máquina, se le denomina "código-objeto", "binarios" o "ficheros ejecutables". En principio, al usuario común sólo le importa este último nivel, los "binarios", pero conviene tener clara la distinción entre fuentes y binarios pues es clave para entender el empeño de los partidarios del software libre en disponer de las fuentes.

Pero el software libre es mucho más que el derecho de los programadores y de los hackers[3] a disponer de las fuentes del código: significa también la libertad de copiar y redistribuir esos programas. Esos derechos, o su ausencia, condicionan a cualquiera que use un ordenador y han configurado la industria del software y de la informática tal y como la conocemos hoy día. También ha dado lugar a un movimiento social —el del software libre— cuya historia reconstruiremos brevemente en las próximas líneas.

2 Los inicios

En la informática de los años sesenta y setenta y en la cultura hacker que surgió en torno a ella, se disponía libremente de las herramientas necesarias y del código fuente de la gran mayoría de los programas. La colaboración forma parte desde antiguo de los hábitos de la comunidad científica y además, ante la diversidad de plataformas, era necesario disponer del código cuando se adquiría el programa para poder *portarlo* al hardware de cada cual. Era tan normal como compartir recetas de cocina y ni siquiera se hablaba de "software libre", pues todo el que quería programar se beneficiaba de ello y veía lógico que los demás se pudiesen beneficiar a su vez. Los hackers copiaban los programas, intercambiaban sus fuentes, podían estudiarlas, evaluarlas, adaptarlas a sus necesidades y a su hardware, reutilizaban una parte del código para hacer nuevos programas... El desarrollo de bienes públicos basados en ese modelo fue exponencial hasta el punto de que gran parte de la tecnología en la que se basa hoy Internet —desde el sistema operativo UNIX hasta los protocolos de red— procede de esos años.

Pero, a principios de los años ochenta, ese modelo entra en crisis, y rápidamente comienza a emerger un modelo privatizador y mercantilista. Los ordenadores, hasta entonces escasos, caros y poco potentes, se hacen asequibles, cada vez más baratos y potentes y aparece un nuevo negocio, el de los productores de software. Los programas se empezaron a vender como productos comerciales independientes de las máquinas y sólo con el código binario, para ocultar las técnicas de

[2] O sea, basado en dos estados, conocidos universalmente como "bits" (*binary digits*). La lógica binaria no es una limitación ontológica de las máquinas, de hecho algunos de los primeros ordenadores, como el ENIAC, operaban en base 10. Si los ordenadores se construyen con arquitectura biestable es porque resultan mucho más sencillos y baratos de fabricar que si ese mismo hardware estuviese obligado a instanciar diez estados distintos.

[3] A lo largo de este artículo usaré el término *hacker* no en el sentido massmediático y distorsionado de "pirata informático", sino en su acepción original, tal y como la define por ejemplo Eric Raymond: "Existe una comunidad, una cultura compartida, de programadores expertos y gurús de redes, cuya historia se puede rastrear décadas atrás, hasta las primeras minicomputadoras de tiempo compartido y los primigenios experimentos de ARPAnet. Los miembros de esta cultura acuñaron el término *hacker*. Los hackers construyeron la Internet. Los hackers hicieron del sistema operativo UNIX lo que es en la actualidad. Los hackers hacen andar USENET. Los hackers hacen que funcione la WWW. Si tú eres parte de esta cultura, si tú has contribuido a ella y otra gente te llama *hacker,* entonces tú eres un hacker."

programación a la competencia. La nueva industria del software comienza a apoyarse en la legislación sobre propiedad intelectual. El mundo UNIX se fragmenta en diversas versiones privatizadas y progresivamente incompatibles entre sí, que los programadores no pueden modificar. Lo que era práctica habitual, se convirtió en un delito: el hacker que compartía el código y cooperaba con otras personas pasó a ser considerado un "pirata".

Al tiempo que los sistemas van haciéndose incompatibles entre sí, la comunidad de investigadores se va desmembrando poco a poco. Muchos hackers ficharon por empresas y firmaron contratos en los que se comprometían a no compartir con nadie de fuera los "secretos de fabricación" (el código fuente). Por su parte, los laboratorios de investigación comenzaron a hacer lo mismo y obligaban a sus hackers a suscribir el mismo tipo de cláusulas. Para cerrar el círculo, los compiladores, los depuradores, los editores y demás herramientas imprescindibles para programar eran propietarios y se vendían a precios respetables: se trataba de que la programación "de verdad" sólo estuviese en manos de la naciente industria de software.

Hubo hackers que no aceptaron esta nueva situación y continuaron con sus prácticas pero parecía solo una cuestión de tiempo que la industria del software propietario arrinconara y dejara definitivamente fuera de la ley la cultura cooperativa y confiada de las primeras comunidades de hackers.[4] Este contexto sirve de base y explica el auge posterior del imperio Microsoft y similares: estaba naciendo el negocio del software propietario y la próspera industria de los ordenadores personales.

3 El proyecto GNU

Son los primeros años ochenta y seguiré la pista de algunos de esos programadores que habían conocido la vieja cultura hacker de los años setenta y que no se plegaron a los designios privatizadores de la industria del software.[5] De hecho, consideraron la privatización un verdadero atentado a los mismos cimientos del proceso de conocimiento. Se cuestiona que la propiedad intelectual sea un derecho natural, y se percibe como una práctica socialmente indeseable.[6]

Con ese planteamiento nace el Proyecto GNU (acrónimo recursivo que significa GNU's Not UNIX, o sea, "GNU No es UNIX") de la mano de Richard M. Stallman, un hacker del emblemático Laboratorio de Inteligencia Artificial del Massachussets Institute Technology (MIT). Era el año 1984, Stallman abandona el MIT para que no interfiera en sus planes y, junto a otros hackers interesados en el proyecto GNU, crea la Free Software Foundation (FSF) en 1985: comienza una labor metódica y discreta, guiada por una asombrosa visión estratégica.[7]

[4] "Muchos programadores están descontentos con la comercialización de software de sistema. Esta puede permitirles ganar más dinero, pero les requiere sentirse en conflicto con otros programadores en general en vez de sentirse como camaradas. El acto fundamental de amistad entre programadores es el compartir programas; Ahora se usan típicamente arreglos de marketing que en esencia prohíben a los programadores tratar a otros como sus amigos. El comprador de software debe escoger entre la amistad y la obediencia a la ley. Naturalmente, muchos deciden que la amistad es más importante. Pero aquellos que creen en la ley a menudo no se sienten bien con ninguna de las dos opciones. Se vuelven cínicos y piensan que la programación es sólo otra forma de hacer dinero." (R. Stallman, *El Manifiesto GNU,* 1985)

[5] "Considero que la regla de oro me obliga a que si me gusta un programa lo deba compartir con otra gente a quien le guste. Los vendedores de software quieren dividir a los usuarios y conquistarlos, haciendo que cada usuario acuerde no compartir su software con otros. Yo me niego a romper mi solidaridad con otros usuarios de esta manera. No puedo en buena conciencia firmar un acuerdo de no divulgación [*nondisclosure agreement*] o un acuerdo de licencia de software." (R. Stallman, *El Manifiesto GNU,* 1985)

[6] "Extraer dinero de los usuarios por un programa con base en la restricción del uso que se le dé es destructivo porque las restricciones reducen la cantidad y las formas en que el programa puede ser utilizado. Esto reduce la cantidad de riqueza que la humanidad deriva del programa. Cuando se opta deliberadamente por restringir, las consecuencias dañinas son destrucción deliberada. La razón por la que un buen ciudadano no utiliza estos medios destructivos para volverse más rico es que, si todos lo hicieran, podríamos empobrecernos todos por la destrucción mutua. Esta es ética kantiana; o la Regla de Oro." (R. Stallman, *El Manifiesto GNU,* 1985)

[7] Esta historia la narra con detalle el propio Stallman en "The GNU Operating System and the Free Software Movement", *Open Sources. Voice from the open source revolution*, O'Reilly & Associates, 1999.

El proyecto GNU se propuso a la sazón una tarea titánica: construir un sistema operativo libre completo. No es sencillo expresar en pocas palabras la enorme dificultad que comporta un proyecto así, sólo al alcance de unas cuantas empresas con miles de programadores a sueldo. No digamos ya si no se dispone de herramientas para hacerlo. Stallman tuvo que empezar casi desde cero, sin modelo bazar, pues no existía la universalizada red Internet tal y como hoy la conocemos; tampoco existía una comunidad de desarrolladores lo suficientemente grande y ni siquiera se disponía de un compilador libre para empezar el trabajo. Una analogía es construir una casa sin disponer apenas de herramientas, por lo que primero hay que fabricarlas: desde picos y palas hasta ladrillos y cemento. Eso sí, contaba con algún material reciclable de "otras casas" —grandes fragmentos de código UNIX y una cultura de reutilizar código—. Stallman y la FSF merecen por tanto un reconocimiento especial en esta historia, pues sin compilador, depurador y editor libres no habría sido posible lo que vino después, incluyendo el propio Linux.

Con todo lo importante que eran esas herramientas, no fue ni mucho menos la principal aportación de la FSF. Y es que los hackers que impulsaron el Proyecto GNU en aquellos años no se conformaron con su trabajo de desarrolladores, ya de por sí formidable. Se dieron cuenta de que necesitaban algo más que crear herramientas de software que dieran libertad a los programadores. Para que el trabajo no fuera estéril y fácilmente reapropiable por intereses privados, precisaban además defender esa libertad en el terreno político y jurídico. El Manifiesto GNU (1985), escrito por el propio Richard Stallman, es la declaración de principios e intenciones del proyecto; inspirada en sus principios, se lanza en 1989 la primera versión de lo que fue posiblemente el mejor logro de la FSF y significativamente no en el terreno informático, sino en el ámbito jurídico: la GPL (General Public License) o Licencia Pública General.[8]

4 La GPL: copyleft para tod@s

Utilizando un brillante juego de palabras, tan del gusto de los hackers, Stallman inventa el concepto de *copyleft,* con el propósito político de garantizar la libre circulación de los saberes contenidos en el software y la posibilidad de que todos contribuyan a su mejora. El copyleft se sirve de las leyes internacionales del copyright para darles la vuelta (*all rights reversed:* "todos los derechos del revés") pues protege el uso en lugar de la propiedad. El autor se reserva los derechos para que su obra pueda ser utilizada por cualquiera con la única condición de que nadie recorte o elimine esos derechos de libre uso: en el momento en que alguien suprima o añada nuevas condiciones que limiten en algo su disponibilidad (por ejemplo, distribuyendo código binario modificado sin posibilidad de acceder a las fuentes modificadas) estaría vulnerando la licencia y perdería el derecho a servirse de ese software. Obligando a transferir esos derechos a cualquiera que copie ese software, lo modifique o no, se beneficia quien está de acuerdo con mantener su futuro trabajo con copyleft, mientras que quien quiera desarrollar software propietario no podrá utilizar código libre y deberá empezar desde cero.

La GPL o Licencia Pública General es la plasmación jurídica del concepto copyleft. Con el tiempo, la GPL se ha convertido en el cimiento del software libre, su baluarte legal, y para muchos constituye un extraordinario ejercicio de *ingeniería jurídica:* con la GPL se asegura que trabajos fruto de la cooperación y de la inteligencia colectiva no dejen nunca de ser bienes públicos libremente disponibles y que cualquier desarrollo derivado de ellos se convierta como por ensalmo en público y libre. La GPL se comporta de un modo "vírico" y, como un rey midas del software, convierte en libre todo lo que toca, es decir, todo lo que se deriva de ella.

Junto al modelo copyleft, hay otros desarrollos de software libre que no son copyleft y considerados menos "estrictos" en cuanto a la licencia, cuya mayor diferencia con el copyleft es que no

[8] Hay varias traducciones no oficiales al castellano de la Licencia Pública General de GNU, por ejemplo en http://lucas.hispalinux.es/Otros/gples/gples.html. Puede leerse la versión original en inglés (única con valor legal) en `http://www.gnu.org/copyleft/gpl.html`.

insisten en que el código derivado tenga que seguir siendo libre. Es el caso de las licencias tipo BSD[9] y las de tipo X11/XFree86: no ponen el énfasis en asegurarse que el software libre siga siéndolo, pues los partidarios de Berkeley consideran que de algún modo eso ya es limitar derechos. Posiblemente, es una postura que se acerca al anticopyright y a la noción de "dominio público" (un bien que jurídicamente no es de nadie), pero es menos comprometida —al menos en cuanto a la licencia— en garantizar que el software libre no deje de serlo. En la práctica y dejando los matices de tipo jurídico, tanto las licencias tipo BSD-XFree86 como la GPL son el baluarte del software libre y ambas representan un referente ético y práctico alternativo al software propietario.

5 Linux

Disponiendo de la GPL y de poderosas herramientas informáticas libres llegamos a los años noventa, con un sistema operativo GNU ya casi completo. Faltaba el *kernel* o núcleo de sistema, una pieza fundamental y muy compleja que se iba retrasando más de lo debido por la enorme dificultad de la empresa y por la escasez de voluntarios que trabajasen en ello (hay que recordar que la mayor parte de los hackers han escrito su código en ratos libres).

Por aquel entonces, por su cuenta y riesgo y sin ninguna relación con la FSF, un estudiante finlandés llamado Linus Torvalds decide ponerse a escribir un kernel que pueda funcionar y sacar todo el partido de la arquitectura de 32 bits de los nuevos procesadores i386. Cuenta con las herramientas GNU para hacerlo, y con un desarrollo UNIX para los PC de 16 bits de entonces (minix). Por primera vez hay máquinas disponibles a nivel personal y a precio asequible capaces de trabajar con un sistema multitarea. Linus decide entonces hacer un llamamiento a través de las news para quien quiera ayudarle en el desarrollo. A los pocos meses (1992), son unos cientos de entusiastas hackers de todo el mundo, coordinados a través del correo electrónico y de las news y sin ningún interés económico, los que consiguen el milagro. A un ritmo frenético y en medio de un caos aparente, van dejando versiones en los repositorios FTP de Internet para que la gente las pruebe, las estudie o las mejore. Linus pone el desarrollo del kernel bajo la GPL y el proyecto GNU se pone a trabajar para integrar el nuevo kernel con el resto del sistema. Desde entonces la historia es bien conocida: a principios del año 2000 son probablemente más de mil hackers los que dan soporte al kernel y se calculan veinte millones de usuarios del conjunto GNU/Linux. En suma, disponemos libre y gratuitamente de un sistema operativo completo, potentísimo y fiable como el que más, que doblega a las grandes firmas y desafía a muy corto plazo al ubicuo imperio de las *Ventanas*™ con miles de programas en constante evolución (la última distribución GNU/Linux del Proyecto Debian[10] recopila más de 4500 paquetes de código libre).

Si bien es el más conocido, el núcleo Linux no es el único ejemplo del increíble éxito del software libre: por ejemplo, el 62% de los servidores web de Internet (equivalente a diez millones de sitios web, según el último estudio de Netcraft,[11] correspondiente a junio del 2000) se basa en un software libre llamado Apache, o en alguna versión modificada del mismo. Apache lo desarrollaron en un tiempo record un grupo de webmasters y ha minimizado el uso de los servidores propietarios de Microsoft y Netscape (en el mismo estudio de Netcraft aparece en segundo lugar, muy lejos de Apache, el IIS de Microsoft con un 20,36% de los servidores web y tercero Netscape-Enterprise con solo un 6,74%). Muchas otras utilidades y aplicaciones basadas en software libre operan en servidores

[9] `http://www.freebsd.org/copyright/`

[10] El Proyecto Debian nació bajo los auspicios de la Free Software Foundation en 1993, con el objetivo de juntar la piezas GNU y construir un sistema operativo libre completo. Hoy día es un proyecto independiente de la FSF pero mantiene sus objetivos fundacionales, lo cual la hace totalmente singular dentro de las distribuciones GNU/Linux: es la única basada exclusivamente en software libre y es la única de carácter no comercial. Debian se mantiene y desarrolla de manera distribuida mediante la cooperación desinteresada de más de 300 hackers de todo el mundo y dispone de una comunidad de miles de usuarios coordinados a través de más de cincuenta listas de correo públicas extraordinariamente activas.

[11] `http://www.netcraft.com/survey/`

de todo el mundo y, de modo silencioso y transparente para el usuario de a pie, garantizan el funcionamiento cotidiano de Internet y de otras muchas redes y sistemas informáticos, libres de los virus y agujeros de seguridad que periódicamente atormentan a los inseguros sistemas basados en Windows. Disponer del código fuente permite localizar errores y corregirlos, e incluso detectar la existencia de código malicioso (virus, puertas traseras, troyanos) que las empresas y grupos de poder pueden eventualmente introducir en los programas y sistemas operativos cerrados como forma de control y de asalto a la privacidad.[12]

El responsable de esta *revuelta* antipropietaria ("Linux es subversivo": así empieza "La catedral y el bazar") no es Linus Torvalds ni Richard Stallman ni la FSF, ni universidad, gobierno o institución alguna, ni menos aún las empresas que ahora cotizan en el Nasdaq con "Linux" como bandera. El responsable de todo esto es la propia comunidad de usuarios del sistema. En el caso de Linus Torvalds, su mayor mérito y por lo que debe ser reconocido sin discusión no es por el kernel Linux, por extraordinario que sea este, sino por el "modelo bazar", la genial intuición de ingeniero que tuvo para ponerlo a tope de vueltas en el momento justo (sin la explosión de Internet y de los ordenadores personales no habría sido posible). Linus ha llevado probablemente hasta sus límites el modelo bazar y lo ha exprimido al máximo, aunque justo es decir que para nada lo inventó pues desde siempre formaba parte de algunos entornos UNIX y de la práctica de determinadas comunidades científicas y académicas (como Bell Labs, el MIT AI Lab o la Universidad de California en Berkeley), que lo aplicaron y obtuvieron éxitos legendarios: pero nadie antes que Linus lo había lanzado a escala planetaria, fuera del ámbito científico y con ese formidable grado de intensidad y productividad. Se puede afirmar sin temor a exagerar que el sistema operativo libre GNU/Linux es la obra más importante que hasta ahora ha producido Internet.

6 El modelo bazar

Actualmente y gracias al proyecto en torno al kernel Linux, el principal modelo de desarrollo del software libre es el "modelo bazar". Fue descrito por Eric S. Raymond en su ya clásico "La catedral y el bazar"[13] (1997) y sin duda constituye una aportación singular en este capitalismo de fin de siglo. Raymond contrapone el modelo bazar a un modelo de producción de software al que denominó "modelo catedral",[14] basado en la necesidad de un arquitecto al mando de un staff rígidamente estructurado y jerarquizado y el estricto control de errores previo a la publicación. A juicio de Raymond, el modelo catedral no sólo corresponde a la industria del software propietario, sino a algunos de los grandes desarrollos libres que ha avalado la FSF.

[12] Un caso paradigmático ha sido el archifamoso gusano LoveLetter (alias "ILoveyou"), que infectó a varios millones de ordenadores conectados a Internet a principios de mayo del 2000. Con un despliegue a partes iguales de sensacionalismo e ignorancia, las portadas de los medios de comunicación de todo el mundo se hicieron eco de este hecho como de un problema que ponía de manifiesto una supuesta falta de seguridad de Internet. Ni ellos ni ninguno de los autodenominados expertos de empresas antivirus señalaron en ningún caso que el "peligrosísimo virus" era un sencillo script de menos de 300 líneas escrito en VisualBasic, inocuo por tanto para las tres cuartas partes de los servidores de Internet, basados en sistemas UNIX. Para eludir toda responsabilidad, Microsoft insistía en que no se trataba de ningún error de diseño en sus aplicaciones. De ese modo y sin advertirlo, Microsoft estaba reconociendo implícitamente que el gusano "I love you" no explotaba agujero alguno, simplemente aprovechaba las facilidades inherentes a la concepción de Windows: es pues Microsoft, y no Internet, el que convierte los PC caseros en absolutamente inseguros y pone los datos de sus incautos usuarios al alcance del más inexperto *script kiddy.*

[13] http://lucas.hispalinux.es/Otros/catedral-bazar/cathedral-es-paper-00.html

[14] Nombre desafortunado para describir el fenómeno, pues la construcción de las catedrales góticas se debía a los *compagnons,* colectivos nómadas e itinerantes del tipo albañiles, carpinteros, herreros, etc. que las construían aquí y a allá, diseminando las obras, sin división entre trabajo manual e intelectual y con una planificación y construcción descentralizada y autónoma: "Al plano sobre el suelo del *compagnon* gótico se opone el plano métrico sobre el papel del arquitecto exterior a la obra." (Gilles Deleuze y Félix Guattari, *Mil mesetas,* Pre-Textos, 1988). Sería pues más exacto denominar "modelo pirámide" o "modelo rascacielos" al modelo jerárquico y planificado que describe Raymond en su artículo.

Según Raymond, el modelo bazar de programación se resume en tres máximas: 1) liberar rápido y a menudo; 2) distribuir responsabilidades y tareas todo lo posible, y 3) ser abierto hasta la promiscuidad para estimular al máximo la cooperación. Incluso cumpliendo esas máximas, no siempre es posible el modelo bazar: sólo puede darse en un entorno de libertad, cooperación, comunidad y disponiendo del código abierto. El bazar encuentra dificultad para producir cooperación cuando se empiezan proyectos desde cero o cuando se ensaya en grupos reducidos demasiado heterogéneos o con mucho desnivel de conocimiento, por lo que a menudo encontramos fórmulas mixtas entre el bazar y la catedral.

A juicio de Raymond, el modelo bazar es mucho más eficaz y produce un software de mayor calidad con menor gasto de recursos, lo que por sí solo ya justificaría la aplicación masiva del modelo en la industria del software. Sin dejar de reconocer esto, la gente que sigue los postulados de la FSF, insiste en que la calidad del código libre ha sido un elemento "extra" y no es la razón de ser del software libre, ya que más importante que la potencia y la fiabilidad técnica es la libertad, el bien social y la comunidad autogestionada de usuarios y desarrolladores a que da lugar, sin precedentes en ningún otro ámbito, que por primera vez lleva la iniciativa y el total control tecnológico sobre lo que usa.

En todo caso, con bazar o sin él y más allá de su demostrado éxito a nivel organizativo y técnico, el software libre desafía la lógica interesada y mercantilista que parecía definitivamente asentada en lo social. Alguien podría objetar que los procesos de cooperación no son una novedad en el capitalismo avanzado y que de hecho son parte imprescindible del modelo de organización posfordista.[15] Pero este último precisa cooperación sujetada, orientada únicamente a la extracción de beneficio, en ningún caso autodeterminada. La novedad que introduce el software libre es que pone en funcionamiento un modelo de *cooperación sin mando.* No hay intereses empresariales directos, es *general intellect* puro, ingobernable y libre del mando.[16] Es más, la ausencia de mando, de control corporativo o jerárquico, parece condición *sine qua non:* allí donde reaparece el mando —sea en forma de interés propietario, sea en su variante autoritaria—, el modelo se marchita, se agosta y acaba por desaparecer. Como el pájaro bobo (el pingüino), sólo puede vivir en libertad. Nadie da órdenes, nadie acepta órdenes. Y sin embargo, la gente se coordina, se organiza, hay gurús, "líderes", gente que dirige proyectos: pero es autoridad conferida, no es mando. Funciona una especie de "economía del regalo", en la cual se es más apreciado cuanto más se aporta a la comunidad. Nadie puede exigir, no hay garantía, no hay dinero como estímulo para el trabajo,[17] aunque haya gente que cobre por su trabajo o gane dinero mediante Linux, pues ninguna objeción hay en la comunidad para que los hackers puedan ser remunerados por su trabajo. Todo este "bazar" caótico de listas y grupos dispersos de voluntarios por Internet produce el mejor software, complejísimo software cuyo desarrollo no está al alcance ni de la empresa más poderosa del planeta. Porque la comunidad del software libre es ya la empresa de software más poderosa del planeta.

[15] En el plano de los procesos productivos y de las formas de mando sobre la cooperación social, algunas corrientes de pensamiento denominan *posfordismo* al conjunto de transformaciones que a partir de mediados de los años setenta conducen a la informatización de lo social, la automatización en las fábricas, el trabajo difuso, la hegemonía creciente del trabajo inmaterial, del *general intellect* y del llamado terciario (comunicativo, cognitivo y científico, performativo, afectivo) y la globalización de los procesos productivos.

[16] En los *Grundrisse,* texto que prefigura nuestra época con más de cien años de antelación, Karl Marx recurre al término *general intellect* (o "intelecto general") para designar el conjunto de los conocimientos abstractos (de "paradigmas epistemológicos", diríamos hoy), que, al mismo tiempo, constituyen el epicentro de la producción social y organizan todo el contexto de la vida. Un "cerebro" o intelecto general, basado en la cooperación y el saber abstracto, incluyendo el saber científico, que tiende a volverse, en virtud precisamente de su autonomía en relación a la producción, ni más ni menos que la principal fuerza productiva, relegando a una posición marginal al trabajo parcelizado y repetitivo de la producción industrial.

[17] "De hecho, mucha gente va a programar sin absolutamente ningún incentivo monetario. La programación tiene una fascinación irresistible para algunas personas, generalmente para las mejores en el ramo." (R. Stallman, *El Manifiesto GNU,* 1985)

7 La teoría de juegos

¿Cómo es esto posible? ¿Por qué ganan las estrategias altruistas a las egoístas en el software libre? ¿Por qué la gente no trata simplemente de extraer el máximo beneficio económico como enseña el capitalismo? ¿Por qué los pragmáticos no se limitan a tratar de aprovecharse y en la práctica cooperan como el que más (aunque ideológicamente no lo reconozcan)?

Desde la propia comunidad del software libre ha habido intentos de explicar estos fenómenos a través de la teoría de los juegos.[18] Y ciertamente, el clásico dilema entre "bien colectivo" *versus* "actitud egoísta" es superado por un axioma que recuerda vagamente al "dilema del prisionero" de la teoría de juegos: la cooperación es preferible *también* desde una perspectiva egoísta. Y, al igual que sucede en el dilema del prisionero, esto no siempre es evidente de primeras. Inventado hace medio siglo por especialistas de la teoría de juegos, el "dilema del prisionero" se utilizó para estudiar el concepto de elección racional y para ilustrar el conflicto existente entre beneficio individual y bien colectivo.[19]

En la teoría de juegos tradicional, la estrategia ganadora es la llamada *Tit for Tat* ("donde las dan las toman"): "sólo coopero si el otro coopera". Es también la más simple, se comienza cooperando en la primera jugada y después simplemente se copia el movimiento previo del otro jugador. Los teóricos de juegos consideran que *Tit for Tat* reúne dos rasgos que identifican a las estrategias ganadoras y que juntas la hacen ganar en todas las pruebas realizadas por ordenador contra estrategias mucho más sofisticadas y mas "sucias" (egoísmo no cooperativo): es *amable* y es *clemente*. Una *estrategia amable* es aquella que nunca es la primera en ser egoísta. Una *estrategia clemente* es la que puede vengarse pero tiene *mala memoria,* es decir, tiende a pasar por alto antiguas ofensas (se venga inmediatamente de un traidor o egoísta, pero después olvida lo pasado). No se olvide que es amable en sentido técnico, no moral, pues no perdona en absoluto. *Tit for Tat* tampoco es "envidiosa", que en la terminología de Robert Axelrod significa que no desean más recompensa que los demás y se siente feliz si el otro tiene el mismo premio que uno mismo (de hecho *Tit for Tat,* nunca gana un juego, como máximo empata con su oponente): en el software libre significa desear que todos tengan las mismas libertades de que dispone uno mismo. Que lo más eficaz sea ser amable y clemente parecía desafiar todo sentido común y constituyó toda una sorpresa para los matemáticos, psicólogos, economistas y biólogos que han estudiado a fondo las diversas estrategias de la teoría de juegos.[20] Esta conclusión, que abrió una nueva dirección de análisis, se ha confirmado una vez tras otra en los estudios y torneos organizados por el politólogo estadounidense Robert Axelrod: siempre acaban ganando las estrategias amables y clementes y siempre salen derrotadas las estrategias "sucias". Por su parte, biólogos, genetistas y etólogos están cada vez más convencidos de que la "cooperación egoísta" es la dominante en la naturaleza.

De acuerdo a la teoría de juegos, los individuos del *Tit for Tat,* cooperando entre sí en acogedores

[18] Ver por ejemplo el artículo de Juan Antonio Martínez, "Software libre: una aproximación desde la teoría de juegos", en *Linux Actual,* núm 11.

[19] Los creadores del "dilema del prisionero" lo ilustraron así: dos personas detenidas y sospechosas de cometer un delito son puestas en celdas separadas e interrogadas. Cada uno es invitado a traicionar a su colega, convirtiéndose en un arrepentido. Lo que suceda depende de lo que hagan ambos prisioneros y ninguno sabe lo que ha dicho el otro. Si los dos se callan (es decir, si cooperan entre sí, según la teoría de juegos), serán condenados a una pena mínima de un año por falta de pruebas. Si se denuncian uno al otro (es decir, no cooperan entre sí, según la teoría de juegos) cumplirán una pena de tres años. Pero si sólo uno denuncia al otro, recibirá una recompensa (y quedará libre), mientras que su cómplice se pudrirá entre rejas durante cinco años. Ante este dilema —suponiendo que ambos están motivados por el interés racional y que no pueden hablarse para pactar entre sí— parece que la única opción racional es acusarse mutuamente para minimizar la pena (será liberado si su cómplice se calla y cumplirá tres años si habla; en cambio pueden caerle cinco años si calla y su cómplice habla). La opción más *racional* les hará acusarse mutuamente y recibir una pena mayor. A menos que el jugador sea un incauto, tendrá que descartar la solución más *deseable* para ambos —la cooperación (o sea permanecer callados)—. Este dilema sin salida ha vuelto locos a generaciones de teóricos de juegos, y solo con una variante llamada el "dilema del prisionero repetido", que consiste en poderlo jugar varias veces y observar el comportamiento del otro, encontraron una condición de salida.

[20] Ver la obra de Richard Dawkins *El gen egoísta,* publicado en su segunda edición en 1989. Especialmente relevante para este asunto es el capítulo "Los buenos chicos acaban primero".

y pequeños enclaves locales, pueden prosperar hasta pasar de pequeñas agregaciones locales a grandes agregaciones locales. Estas agregaciones pueden crecer tanto que se extiendan a otras áreas hasta entonces dominadas, numéricamente, por individuos egoístas que juegan al "Voy a lo mío". A su vez, la cooperación es un fenómeno que produce realimentación positiva: nadie que disfrute de los beneficios del software libre puede dejar de promover su uso. Por eso la comunidad conserva cierto tono proselitista, además de por una percepción más o menos generalizada de que la potencia y el futuro del modelo depende muy directamente de que haya mucha gente participando activamente en su desarrollo.[21]

Sin embargo, el modelo *Tit for Tat* no caracteriza totalmente al software libre, al menos no de una manera canónica. Por un lado, es libre incluso para quienes no cooperan (esto le da valor ético, pero le aleja del *Tit for Tat*). Por otro lado, aunque el copyleft permite que cualquiera se beneficie, no permite que nadie se lo apropie o que se use para crear software propietario (esto le da valor pragmático y le aproxima al *Tit for Tat*). La estrategia del software libre es "amable" y "clemente" a la vez, pero —a diferencia del *Tit for Tat*— es capaz de asumir en su seno estrategias egoístas sin necesidad de expulsarlas o vengarse de ellas (salvo quizá en casos en que se percibe un verdadero peligro, como que alguna empresa poderosa adoptase posiciones descaradamente egoístas no cooperativas, por ejemplo vulnerando la GPL).

En el software libre, convive un planteamiento basado exclusivamente en la eficacia, en la superioridad técnica y productiva que genera el modelo bazar, con otro que sitúa en primer plano la cooperación, la ética y la libertad. La postura pragmática, que hay quien ha calificado críticamente como "realpolitik"[22], rechaza cualquier formulación ética del modelo y sólo acepta como ideología las reglas del *laissez-faire* propias del liberalismo más ortodoxo. Este planteamiento apoya y potencia decididamente el software libre, porque ha verificado que su resultado es *más eficaz,* no porque valore la cooperación en términos de producción de bienes públicos o de beneficio social ni porque considere inmoral el modelo propietario.[23] Incluso puede que solo le interese la cooperación social como poderosa máquina al servicio del capitalismo. Esta parte del modelo probablemente es la que se ajusta mejor a la teoría de juegos de acuerdo al concepto de la *cooperación egoísta:* las empresas cooperan porque a la larga obtendrán más beneficios y los individuos cooperan porque apoyando el modelo dispondrán de mejores aplicaciones.

Junto a este planteamiento coexiste un acercamiento ético o altruista. Conviene no confundirlo con un altruismo de base moral, religiosa o metafísica, sino de una ética materialista que considera la libertad y la cooperación social el mejor modo de defender algo que es bueno para todos y que encuentra otros estímulos diferentes al beneficio económico.[24] Dicho de otro modo, no se trata de una historia de "altruistas" y "egoístas", de "buenos" y "malos", que como tantos otros dilemas morales se han mostrado inoperantes por falsos: pero hay una cuestión política de fondo muy importante

[21] Juan Antonio Martínez, *op. cit.*

[22] "Todas las confusiones y parcialidades que aparecen en los artículos de Eric Raymond son típicos de su elección de la 'política real' como principio de actuación en su activismo en pro del software libre. Un ejemplo de esta elección es haber cambiado con efectos retroactivos en sus artículos y conferencias el término *software libre* por *open source.* No discrepo de la noción de ser eficaz promoviendo el software libre. Pero me opongo a acciones que pueden resultar atajos válidos a corto plazo y causar perjuicios a la larga, ya que en estos casos, en la búsqueda de un éxito puntual, se opta por apoyar fenómenos esencialmente erróneos en lugar de combatirlos." (François René Rideau, "Sobre los artículos de Eric S. Raymond", 1998)

[23] "Es posible que a largo plazo triunfe la cultura del software libre, no porque la cooperación es moralmente correcta o porque la 'apropiación' del software es moralmente incorrecta (suponiendo que se crea realmente en esto último, lo cual no es cierto ni para Linus ni para mí), sino simplemente porque el mundo comercial no puede ganar una carrera de armamentos evolutiva a las comunidades de software libre, que pueden poner mayores órdenes de magnitud de tiempo calificado en un problema que cualquier compañía." (Eric Raymond, "La catedral y el bazar", 1997)

[24] "No hay escasez de músicos profesionales que sigan en lo suyo aunque no tengan esperanzas de ganarse la vida de esta forma. [...] Durante más de diez años, varios de los mejores programadores del mundo trabajaron en el Laboratorio de Inteligencia Artificial [del MIT] por mucho menos dinero del que podían ganar en otras partes. Ellos obtenían varios tipos de premios no monetarios: fama y aprecio, por ejemplo. Y la creatividad también se disfruta, es un premio en sí mismo." (Richard Stallman, *El Manifiesto GNU,* 1985)

que los diferencia claramente y es la de si el software —y, en general, el saber humano— puede o no puede ser privatizado. Mientras para el sector pragmático esto no es relevante, para Stallman y quienes abogan por la visión ética esto es una cuestión central e innegociable: el software, a diferencia de los bienes materiales, no puede ser poseído, pues puede ser disfrutado por un número indeterminado de personas sin que por ello haya que privar a nadie de tenerlo a su vez.[25] Ese es el núcleo del dilema, de la diferencia, y el que comporta acercamientos tan dispares al software libre.

La teoría de juegos funciona a nivel estadístico y se basa en estrategias inconscientes de base algorítmica (las pueden ejecutar máquinas, genes o seres humanos): no aplica pues criterios morales o finalistas ni trata de dar cuenta de los casos particulares, ni de las motivaciones de cada cual para cooperar o para ser egoísta, sino que nos ofrece algo más sutil y valioso: la comprensión de un proceso y el cuestionamiento de un mito capitalista y neoliberal, el del juego sucio y el "todos contra todos", el de que es mejor que cada uno vaya a lo suyo y solo se atienda a los intereses privados. Las conclusiones de la teoría de juegos —pese a carecer de finalidad moral— nos ofrece un resultado optimista y alentador para una ética materialista (no moralista ni religiosa). La teoría de juegos y el software libre podrían ser la punta de lanza de un *nuevo mito,* el mito de compartir, el de la cooperación y la ayuda mutua. Podría anunciar la saludable idea de que incluso con individuos egoístas al mando, y en palabras del biólogo Dawkins, "los buenos chicos acaban primero".

No obstante, hay también razones para pensar que si el enfoque pragmático, apolítico, también llamado "cooperación egoísta", se acaba imponiendo, dañará a la comunidad del software libre, que podría acabar siendo recuperada por el capitalismo posfordista, del mismo modo que recupera el *general intellect* (la cooperación y el saber social general) y lo pone al servicio de la extracción de beneficio privado. Otros, sin embargo, apuestan por la coexistencia de ambas tendencias, y piensan que mientras la postura egoísta se avenga a cooperar dentro de las reglas del software libre no habrá nada que temer. Ese debate lo abordaré en el siguiente epígrafe.

8 Desafíos e interrogantes

> *El interés en el software crece más rápido que la conciencia acerca de la filosofía sobre la cual está basado, y esto crea problemas. Nuestra capacidad de enfrentar los desafíos y amenazas al software libre depende de la voluntad de mantenerse firmes del lado de la libertad. Para asegurarnos de que nuestra comunidad tiene esta voluntad, necesitamos esparcir la idea entre los nuevos usuarios a medida que ellos llegan a nuestra comunidad. Pero estamos fracasando en esto: los esfuerzos realizados para atraer nuevos usuarios a nuestra comunidad sobrepasan de lejos a los esfuerzos dedicados a la enseñanza cívica acerca de nuestra comunidad. Necesitamos hacer ambas cosas, y es necesario que mantengamos ambos esfuerzos equilibrados.*
>
> Richard Stallman

Entre 1997 y 1998 se producen tres acontecimientos que, a juicio de muchos, inauguran un nuevo periodo en el ámbito del software libre: la publicación de "La catedral y el bazar"; la liberación del código fuente de Netscape, y la foto de Linus Torvalds en la portada de la revista *Forbes.* Convencionalmente, se considera que esos tres hitos despertaron el interés y dieron pie a la entrada masiva de las grandes empresas en el mundo del software libre. Esta nueva etapa está llena de sombríos interrogantes, por mucho que algunos la describan triunfalmente, y probablemente van a generar nuevos focos de antagonismo. Algunas grandes empresas han comenzado a contratar hackers (lo cual no es nuevo) para llevar a cabo desarrollos de software libre (esto sí lo es). Trabajos que antes se hacían sin interés económico directo ahora empiezan a estar financiados por empresas. Proyectos cuya motivación era la necesidad o el deseo de los hackers y de la comunidad de usuarios de software libre, ajena al mercado, ahora pueden empezar a estar condicionados por las necesidades,

[25] "Como no me gustan las consecuencias que resultan si todos acapararan información, debo considerar como erróneo que alguien lo haga. Específicamente, el deseo de ser recompensado por la creatividad de uno no justifica el privar al mundo en general de toda o parte de esa creatividad." (Richard Stallman, *El Manifiesto GNU,* 1985).

los ritmos y las prioridades de las empresas que financian esos proyectos.[26] Modestos negocios que basaban sus ingresos en servicios relacionados con el software libre se han convertido de la noche a la mañana en grandes empresas que han salido a bolsa con capital-riesgo. Algunas empresas que basan su negocio en el software libre se están dedicando a comprar empresas más pequeñas y a su vez son compradas por otras mayores, produciéndose la creación de grandes emporios. Ese trajín de compraventa incluye sitios estratégicos para la comunidad como medios de comunicación o repositorios de software: Andover compra Slashdot y Freshmeat; VA Linux compra Andover; RedHat compra Cygnus, etc. ¿Adónde conduce esa concentración empresarial? ¿Qué pinta la gente de a pie en todo este tinglado?

Hasta ahora, en la comunidad del software libre todo esto no se aprecia como una amenaza, ni siquiera como un problema, antes al contrario: alguna gente se ha esforzado mucho para convencer a las empresas de la viabilidad capitalista del modelo, y ahora empiezan a recogerse los frutos. ¿Cómo vamos a oponernos ahora a que las empresas ganen dinero con el modelo, siempre y cuando mantengan las reglas del juego, es decir, produzcan o financien software libre?

Ni tenemos perspectiva ni ha pasado tiempo suficiente (apenas dos años) para valorar lo que va a suponer la irrupción masiva de capital fuerte y de transnacionales en el software libre. Mi apreciación personal es que, a diferencia de otras cuestiones en que se mantiene una actitud crítica y muy alerta (como la legislación sobre patentes), en este crucial asunto hay excesiva fe en las bondades del mercado y del libre comercio. Es cierto que hasta ahora se ha conseguido doblegar a verdaderos gigantes, pero ahora la situación es distinta porque con el modelo de "cooperación egoísta" —y qué mayor paradigma de la cooperación egoísta que el de la empresa capitalista— esas empresas juegan a estar "dentro". Se puede pasar fácilmente de la cooperación sin mando a la cooperación sujetada, la cooperación con mando.

Se presupone, en contra de toda evidencia histórica anterior, que lo que es bueno para las empresas es también bueno para las personas. Y el axioma no es ese, sino uno más tautológico pero también más exacto: *lo que es bueno para las empresas es bueno para las empresas.* Y nada más. Bien es cierto que, a veces, aquello que genera beneficio empresarial es reutilizado para procurar beneficios sociales, pero esto es colateral (como un epifenómeno) y es atrozmente ingenuo confiar a priori en que va a ser así. La confusión entre lo que es bueno para las empresas (la acumulación de capital y la extracción de beneficio económico por encima de cualquier otra consideración) y lo que es bueno para la gente (la producción de bienes públicos y de riqueza social para la vida en comunidad) puede ser desastrosa. *Todo el interés del capitalismo en el software libre es convertirlo en una máquina más de hacer dinero,* pero como con todo lo demás si lo consigue probablemente será a costa de vaciarlo de todo contenido liberador.

El sector que va más allá de la superioridad técnica y que realiza una apuesta por la dimensión ética del software libre, confía en la fortaleza del movimiento y de momento no se percibe alarma alguna en este sentido. Se considera que el modelo de producción del software libre no puede ser privatizado y recuperado por el mercado, que está blindado *jurídicamente* (la GPL), *técnicamente* (la superioridad en varios órdenes de magnitud de lo creado mediante el modelo bazar frente a sistemas propietarios) y *políticamente* (algunos de los más significativos promotores del software libre provienen de movimientos contraculturales o simpatizan con causas pro derechos civiles).[27]

No obstante y compartiendo esa confianza en la potencia de la comunidad y su capacidad de respuesta, demostrada ampliamente hasta ahora, no hay razón para desechar una lectura más

[26] Un admirado hacker, que coordina un estratégico proyecto de software libre, me comentaba en privado recientemente que hasta hace un año se levantaba por la mañana y se ponía a escribir lo que le apetecía o si no le apetecía no escribía nada. Ahora, en su empresa, él sigue trabajando con software libre pero cuando se levanta por la mañana debe consultar su agenda y ponerse a escribir lo que le piden sus clientes. Aunque en ambos casos, antes y ahora, está produciendo software libre, la diferencia a su juicio es muy notable.

[27] Ver artículo de Aris Papathéodorou y Laurent Moineau, "Coopération et production immaterielle dans le logiciel libre", en el primer número de la revista *Multitudes*, marzo del 2000. Hay disponible una versión en línea en: `http://www.samizdat.net/slut/textes/mult_coopprod.html`.

crítica que nos haga cuando menos estar alerta y no relajarnos ante los éxitos y los cantos de sirena que vienen de fuera: el capitalismo ha sido capaz de "recuperar", privatizar y mercantilizar casi todos los aspectos de la producción y de la vida, desde lo material a lo inmaterial. ¿Por qué no va a poder hacer lo mismo con el software libre? De hecho, hay ya bastantes indicios que apuntan a la "recuperación" mercantilista de la capacidad de innovación del hacker. El asalto masivo de las grandes empresas, con perspectiva exclusivamente mercantil, podría verse como un "troyano"[28] introducido en el software libre y que, con el tiempo, parasite y desactive la potencia de la comunidad. ¿De qué modo? La confusión con el tema de las licencias, por ejemplo, está debilitando progresivamente la filosofía de fondo del software libre (por eso la FSF se esfuerza tanto en explicar las diferencias entre unas y otras), haciendo que algunas empresas hagan pasar por libres desarrollos que no lo son, o bien popularizando distribuciones comerciales "Linux" que mezclan software propietario con la base libre del sistema GNU/Linux. Lo primero —la confusión con las licencias— puede causar desconfianza entre los desarrolladores, que teman que su trabajo pueda ser finalmente reapropiado y privatizado, y lo segundo —las distribuciones GNU/Linux que añaden software no libre— tapona el desarrollo de opciones libres que reemplacen esas soluciones propietarias y se legitima software propietario como si por el hecho de funcionar bajo GNU/Linux fuese "menos propietario" y más aceptable. Por su parte, la Open Source Initiative (OSI) no ha ayudado mucho a aclarar este panorama. Surgió como propuesta de algunos hackers para acabar con una ambigüedad (*free* en inglés, significa "libre" pero también "gratis") y con un término que al parecer disuadía a las empresas, pero a cambio ha introducido otras tal vez peores: con el concepto *open source* ("fuente abierta") que proponen como sustituto a "software libre" se pone solo el acento en que el código fuente esté disponible, sin incidir en las otras dos libertades (poder copiar y poder redistribuir libremente). Es decir para solucionar una ambigüedad, se ha creado otra mayor.

Otro problema derivado del *troyano* de la mercantilización son los agravios comparativos que pueden producirse entre hackers que cobran de multinacionales mucho dinero por el mismo trabajo y proyecto en que otros participan sin cobrar. También hemos citado el peligro de que las empresas marquen las prioridades de desarrollo y que se privatice el conocimiento. Esto último —la privatización del conocimiento— entronca con dos de los problemas más graves con el que se debe medir el software libre: 1) la poca o nula disponibilidad de los fabricantes a facilitar información técnica relevante sobre sus dispositivos ni a fabricar drivers para GNU/Linux que permitan utilizar los nuevos dispositivos que van apareciendo en el mercado, y 2) las patentes del software, como forma de privatización de las ideas, verdadera amenaza para el software libre, ya que obligan a esperar durante años a que expiren patentes de invención que son cruciales para poder utilizar determinados programas.

Algunas de esos elementos, o varios combinados entre sí, podrían desmoronar la cooperación sin mando y, por tanto, la comunidad de software libre tal y como hoy la conocemos: y si no hay comunidad, no hay software libre; puede haber fuentes abiertas y públicas incluso, pero no software libre. Se hace pues cada vez más necesario un análisis político del software libre que lleve a una toma de postura política o, si se prefiere, a una apuesta ética que no ponga en primer lugar la conveniencia o la mera instrumentalización de si es mejor o peor que las opciones propietarias. Estamos ante un fenómeno que escapa claramente a los parámetros clásicos de la economía política y de la ideología: escapa a los parámetros ideológicos al uso, pues ni acaba de encajar en una visión antagonista —hay grandes dosis de pragmatismo y no existe una visión decididamente anticapitalista— y tampoco encaja en el neoliberalismo puro y duro —la libertad absoluta es un valor fundamental del movimiento, sí, pero no el único pues hay también principios éticos acerca de lo público, del apoyo mutuo y del acceso igualitario y horizontal a los recursos del conocimiento y en contra de la

[28] No es solo una metáfora: un "troyano", en este contexto, sería un tipo particular de *meme*. El *meme* sería la idea del mercado como motor de la economía y dinamizador del software libre. En esa hipótesis, el troyano incorporado en el *meme* lo "vampiriza", se propaga con él, y acabaría reduciendo el software libre a una pieza más del gran supermercado global en que se está convirtiendo gran parte de Internet.

privatización del saber humano—. Es una nueva noción de bien público, no tutelado por el mercado ni por el Estado: es un nuevo *espacio público no estatal.* No hay duda de que un nuevo modelo de cooperación social productiva ha surgido en torno al software libre: falta saber lo que dará de sí esa comunidad, además de buenas herramientas informáticas, y si este nuevo paradigma podrá extenderse a otros sectores de la producción inmaterial. Estamos pues ante una verdadera contienda política, que no está ganada ni mucho menos, y que requiere determinación y apoyo al software libre y una lucha decidida contra las patentes de software y demás leyes sobre la propiedad intelectual que previsiblemente podrían detener su avance.[29] Me gustaría acabar citando unas palabras de los paleoantropólogos Carbonell y Sala, del proyecto Atapuerca, pues me parecen un magnífico colofón que de algún modo resume y explica dónde reside la singularidad y la potencia del software libre: "*No es la humanización de la tecnología lo que debemos buscar, sino su socialización. No es posible humanizar algo que es exclusivamente humano. La socialización es lo que permite un crecimiento exponencial de las capacidades humanas.*"[30]

9 Reconocimientos

Sin la propuesta primero y la insistencia después de la Oficina 2004 de Barcelona para escribir esta introducción política al software libre, probablemente nunca me habría puesto a ello. Al primero que escuché explicar con claridad la diferencia entre el acercamiento ético y el pragmático al software libre fue a Jesús González-Barahona. Buena parte de las ideas expresadas en la sección "Desafíos e interrogantes" son fruto de mis frecuentes conversaciones con Marga Padilla, de quien tanto he aprendido a lo largo de muchas horas de *hackin'* —y de vida— compartidas durante estos tres últimos años. Agradezco también la atenta lectura que Luis Pueyo hizo de la versión preliminar de este artículo, cuyas críticas, comentarios y sutiles observaciones sin duda han mejorado el texto final. Por último, *but not least,* la confianza y el apoyo del CSOA el Laboratorio a la telemática antagonista me ha permitido aprender a socializar el conocimiento y ha demostrado que son posibles en un centro social okupado proyectos estables de autoproducción basados en nuevas tecnologías.

Lavapiés, Madrid
Julio del 2000

[29] "Las leyes de propiedad intelectual han corrompido completamente las instituciones económicas, ya que muchas corporaciones dependen de un modo crucial del monopolio de la información, también en el origen de grandes fortunas, más que de la prestación de servicios reales. No deben realizarse compromisos con estas leyes, y debe lucharse contra su justificación, habitualmente errónea. Este es el principio fundamental de la filosofía del software libre. [...] Es difícil luchar contra prejuicios que sirven para justificar enormes intereses financieros, por supuesto. Hace falta ser muy estricto precisamente porque la tarea es muy dura." (François René Rideau, "Sobre los artículos de Eric S. Raymond", 1998)

[30] Eudald Carbonell, Robert Sala, *Planeta humà,* Editorial Empúries, 2000.