EL SOFTWARE LIBRE DESDE EL PUNTO DE VISTA DE LAS FILOSOFÍAS DE LA MULTIPLICIDAD: DINÁMICAS DE COOPERACIÓN Y TRABAJO DEL MODELO BAZAR COMO MODELO ALTERNATIVO AL CAPITALISMO HEGEMÓNICO

Germán Martín Dartsch
Universidad Nacional de Cuyo (Argentina)

1. Una breve historia de la cultura hacker

La cultura hacker como la conocemos surge en el laboratorio de Inteligencia Artificial del Instituto de Tecnología de Massachusetts (Massachusetts Institute of Tecnology - MIT). En 1961, el Club de Tecnología de Trenes a Escala del MIT empezaba a experimentar con la primera computadora PDP 10 que el Instituto adquirió. Son ellos quienes empezaron a utilizar la palabra “hacker”. Los hackers comenzaron a experimentar con la computadora a la vez que exploraban nuevas aplicaciones que extendieron el uso que hasta entonces se daba a esta tecnología. Un ejemplo de esto son los distintos intentos, que finalmente tuvieron éxito, de crear algo que a muchos les parecía imposible: un programa capaz de jugar al ajedrez. Hoy, estos programas son algo común que no sorprende a nadie (Gradín, 2004: 9-13).

Poco tiempo después, los aficionados a la computación surgidos del club de trenes conformaron el núcleo del nuevo laboratorio de Inteligencia Artificial (1). A partir de este momento, se constituyó una comunidad que hizo muchos aportes a la informática y que trabajaba cooperando entre sí horizontalmente solo por el entusiasmo y la afición que la informática les despertaba. En 1969, el laboratorio de IA del MIT pasa a ser conocido y la cultura hacker se expande gracias a la conexión que entre sí pudieron establecer los investigadores en informática al surgir la red ARPANET, antecesora de Internet.

La cultura hacker que se desarrolló en el MIT estuvo signada por la tecnología que utilizaron sus impulsores. Las computadoras PDP 10, desarrolladas por la empresa DEC, eran computadoras de recursos compartidos, es decir que las herramientas informáticas tanto de hardware como de software de una computadora podían ser utilizadas dentro de una red por otras computadoras conectadas. Esto hacía que la cooperación entre los hackers del MIT fuera algo común e incluso necesario. Los hackers del MIT descartaron el software que venía con la computadora y crearon el ITS, que significa Sistema de Recursos Compartidos No Compatible. Las PDP 10, el ITS y la red ARPA fueron el contexto en el que la cultura hacker creció durante los setenta no solo en el MIT, sino también en otros centros de investigación cuyos hackers se comunicaban entre sí a través de ARPANET. Fue una época de importantes avances técnicos e innovaciones.

Sin embargo, ITS, al igual que todos los sistemas operativos de aquél entonces, no era portable, es decir, solamente funcionaba dentro de la computadora para la cual había sido creado (la PDP 10 en este caso). Esto eventualmente haría entrar en crisis a la cultura del IA Lab del MIT. En 1969, Ken Thompson de los laboratorios Bell creó Unix, el primer sistema operativo portable que estaba programado completamente en lenguaje C, un lenguaje genérico utilizado en todas las computadoras (antes de esto, se programaba en el lenguaje assembler,

que requería una programación específica según el hardware en el que se implementara). Con Unix se incorpora la posibilidad de programar directamente en la superficie del programa informático sin necesidad de trabajar sobre el código binario (lo que suponía un gran ahorro de tiempo). En 1974, el sistema había demostrado su portabilidad excelentemente, además de una gran simpleza en su uso. Se empezaba a gestar la computación para los legos. En 1975, también sale al mercado la primera computadora personal (PC), pequeña y barata, lo que permitió el ingreso de muchas personas al mundo de la informática y a ARPANET.

En auge estos cambios, al iniciar la década de los ochenta la comunidad del MIT se disuelve. La PDP 10 estaba desapareciendo, desplazada por las nuevas computadoras PDP 11 y la línea VAX. Al desaparecer la PDP 10, el ITS, dada su no portabilidad, quedó obsoleto y muchos de los hackers del MIT abandonaron el lugar para trabajar en corporaciones de software. Ante esto, y dado el avance cada vez más vertiginoso de las computadoras personales, inicia su mejor momento el software propietario, que impone lógicas restrictivas sobre el código fuente (2). Compartir el código fuente era una práctica común entre los hackers, pues siendo todos entendidos en la informática podían utilizarlo para mejorar el programa a su gusto y personalizarlo. Pero con la principal comunidad de hackers disuelta y un ingreso cada vez más masivo de no expertos al mercado del software, la restricción del acceso al código fuente se generalizó y se volvió hegemónica.

Obviamente, no todos los hackers vieron con buenos ojos este cambio. Para el año 1982, sobre la base de Unix, Richard Stallman (exinvestigador del MIT) empezó a desarrollar GNU, un sistema operativo libre. Al tener éxito varios de sus programas libres y al obtener la cooperación de muchos hackers, creó la Fundación para el Software Libre (Free Software Foundation), soñando con generalizar la libertad de cooperación y la práctica de compartir el software junto con su código fuente.

En la misma época se creó el módem Motorola 68000 que, junto a los protocolos de internet de Unix, se constituyeron en el estándar de navegación de Internet, que comienza así a generalizarse y a usarse masivamente.

En 1984, Unix se vuelve comercial por primera vez y para inicios de los noventa aparecen las computadoras personales basadas en el chip Intel 386, baratas pero a la vez tan potentes como las grandes computadoras de antes. Con este paso, la informática se generaliza definitivamente. Sin embargo, el abaratamiento de las computadoras no fue acompañado por una reducción en el precio de Unix, cuyo costo era excesivo en relación con la tecnología a la que acompañaba. Así, en la década de los noventa una gran variedad de versiones de Unix se estancaron compitiendo entre sí, y sus disputas legales le dejaron el camino abierto a Windows para imponer su hegemonía en el mercado.

Mientras tanto, GNU aún no estaba listo. Faltaba el centro del sistema, el kernel, es decir, centro de instrucciones más importante del sistema operativo. Recién en 1991 el estudiante finlandés Linus Torvalds empezó a desarrollar un kernel libre basado en Unix para las computadoras personales 386 al que llamó Linux. Dado que el trabajo necesario para terminarlo era inmenso, este hacker tomó la costumbre de publicar con mucha frecuencia sus

resultados en Internet y permitir que cualquier interesado pudiera introducir mejoras, para luego publicar la nueva versión. Dio así impulso a un modelo de cooperación libre, descentralizada y sin mando que Eric Raymond denominó “modelo bazar” (Raymond, 2009). Esto se logró gracias a la proliferación de Internet y la popularización de las computadoras personales, que hicieron posible que un gran contingente de entusiastas de la informática tuviera contacto entre sí a escala global, pero también fue muy importante la influencia del proyecto GNU y sus antecedentes para que se concibiera esta lógica de trabajo. Para fines de los noventa, se unieron el Kernel Linux y los desarrollos de GNU para dar como resultado GNU/Linux, el primer sistema operativo capaz de competir con Unix y, en opinión de muchísimos hackers y expertos como Eric Raymond, muy superior a Windows en varios aspectos técnicos como la estabilidad, la asignación de recursos, la protección contra virus informáticos, entre otros ítems menos importantes.

Sin embargo, la serie de disputas y tensiones entre el software libre y el software propietario solo había comenzado y hoy en día sigue siendo el software propietario el que domina entre los usuarios comunes, por lo que podríamos decir que la socialización del software todavía está en proceso; en ese entonces, la lucha de los hackers recién comenzaba. Una vez generalizada Internet, la figura del hacker adquirió fama pública y surgieron también muchos movimientos de intervención política. Esto también suscitó, como sucede con todo movimiento social disidente, una lucha por las palabras en torno al significado del término *hacker*, que pasó a relacionarse y confundirse con la piratería informática por parte de quienes buscaron desacreditar al software libre y sus partidarios.

2. Software libre

Como acabamos de ver, el acto de compartir software no es algo nuevo, lo nuevo es que la cooperación se dé a través de la red, a través del modelo bazar que posibilitó Internet. Como narramos, era normal en los inicios de la informática que el software se compartiera libremente entre los hackers, como la comunidad del MIT lo hacía, así como también era común la práctica de modificar y reutilizar el código de los programas ya existentes para crear versiones mejoradas o programas nuevos. Estos resultados también se compartían. Dijimos ya que es en los años ochenta cuando se impone un modelo privatizador y el viejo modelo entra en crisis. Se crean así instrumentos legales para imponer al software el modelo de la escasez, haciendo imposibles las prácticas de compartir y cooperar (3).

Puesto que en esta década los ordenadores personales estaban empezando a conocer su auge, resultaba fácil adquirirlos y la informática se generalizaba a públicos que antes no tenían acceso a ella. El negocio del software comercial empezaba a formarse bajo las lógicas de cualquier negocio capitalista. Las políticas del copyright y las patentes permitieron imponer la lógica de la escasez al software mediante mecanismos jurídicos. Para adquirir programas era ahora necesario aceptar contratos que prohibían la práctica de compartir el software. Al mismo tiempo, los nuevos productos de software ya no se distribuían junto a su código fuente, por lo tanto era imposible modificarlos y estudiarlos. Otro problema que enfrentaron los hackers fue la

privatización de las herramientas básicas para llevar adelante la tarea de programar software. Además de todo esto, los laboratorios de investigación de las universidades empezaron a tomar la misma actitud que las empresas y a suscribir a las políticas de copyright y confidencialidad. De esta forma, los hackers vieron cómo su comunidad se desmembraba y desaparecía, asediada por limitaciones por todas partes e imposibilitada de seguir con sus costumbres habituales. En este clima hubo muchas y muy tentadoras ofertas de trabajo para los hackers, quienes quedaban sin muchas otras posibilidades que adaptarse a los cambios y seguir las nuevas reglas de juego.

Pero, como ya contamos, hubo quienes no se resignaron de inmediato a abandonar todas las costumbres y cultura que habían creado durante los años sesenta y setenta. Fue así como se inició el movimiento del software libre como respuesta a esta realidad. Hackers como Stallman decidieron buscarle la vuelta al asunto y seguir adelante con la práctica de compartir y modificar el software. Ahora bien, definamos lo que significa software libre. Según Stallman, el software libre es un software que da al usuario libertad (hace esta aclaración por el equívoco que genera la palabra libre en inglés: "free" puede significar tanto libre como gratis). El software libre bien puede ser distribuido gratuitamente como vendido, pero lo que debe permanecer siempre inalterado son las libertades de utilizar, compartir y modificar el software, tal como existían antes del movimiento privatizador de los ochenta. En palabras de Stallman, existen cuatro libertades que definen al software libre:

> "Libertad 0: la libertad para ejecutar el programa sea cual sea nuestro propósito. Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades. Libertad 2: la libertad para redistribuir copias y ayudar así a tu vecino. Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad" (Stallman, 2004: 45).

Decimos entonces que el software libre supone la libertad de compartir el software y de modificarlo, para lo cual es necesario que se lo proporcione junto a su código fuente. Para entender esta lógica que pone en primer lugar el libre acceso a las condiciones que permiten el estudio y desarrollo del software debemos recordar que los orígenes de la cultura hacker se hallan en el MIT y otros institutos universitarios. Los hackers no surgen del mundo comercial, sino del mundo académico (los laboratorios de investigación de las universidades como el MIT). Robert Merton postulaba que el comunismo científico es una de las bases del desarrollo de la ciencia (Merton, 1977: 355-368). Con esto quería decir que los sucesivos desarrollos de la ciencia (y de la filosofía, y tal vez de las ideas en general) no hubieran sido posibles de no ser por la posibilidad de tomar libremente ideas que no nos pertenecen y adaptarlas a nuestros propios intereses, reutilizarlas, modificarlas y crear nuevas ideas en base a las anteriores, siempre y cuando citemos la fuente que estamos retomando. Por ejemplo, para redactar este ensayo yo estoy utilizando ideas y argumentos de otros autores sin tener que pedir permiso ni pagar sumas de dinero para usarlas, solo basta con cumplir los estándares de citación, parafraseo y atribución de las ideas a sus autores originales. Esto es cooperación libre entre

cerebros, tan antigua como el pensamiento mismo. Si no fuera así, no habría posibilidad de avanzar o el mundo académico sería otra cosa completamente distinta. Hasta los años ochenta, en los laboratorios en los que se investigaba software la realidad era la misma. Por lo tanto, la problemática del software libre va más allá de sí misma. Podríamos decir que lo que está en juego es la privatización o libertad del intelecto humano general o general intellect. El concepto de general intellect fue utilizado por primera vez por Karl Marx y se refiere a todas las facultades y disposiciones genéricas del intelecto humano, un intelecto social que dadas ciertas condiciones pasa a ser el principal insumo de producción de capital (Marx, 1972: 216-230). Este intelecto general comprende conocimientos comunes a la humanidad tales como elementos culturales, el lenguaje y sus formas lógicas, las vivencias del cuerpo humano, el sentido común, vivencias preindividuales e incluso las capacidades mismas de conocer y aprender. Hoy este concepto es muy utilizado por una serie de autores que teorizan sobre el llamado *capitalismo cognitivo,* etapa actual del capitalismo en la cual el conocimiento pasa a ser la base directa de la producción de bienes materiales e inmateriales, ya sea directa o indirectamente. Al introducir el concepto de general intellect podemos entender mejor cuál es la base de la dinámica del modelo bazar que mencionamos ya en reiteradas ocasiones. ¿Cómo explicar que en el modelo bazar convergen incontables individuos en un proyecto concreto a través de un medio tan disperso como lo es Internet? La respuesta es porque poseen una base común, una serie de facultades y saberes comunes sobre las que se puede edificar nuevo conocimiento, nuevos productos creados en conjunto y sin la necesidad de una dirección centralizada. Esta es, en nuestra opinión, la base y el punto clave de la cooperación libre en el modelo bazar: la base común del general intellect.

3. Modelo Bazar

Ahora bien, hemos mencionado el modelo bazar desde el inicio del ensayo, pero aún no hemos precisado a qué nos referimos exactamente. El modelo bazar de desarrollo de software existía antes de que se empezara a crear Linux, era el modelo de trabajo de las comunidades de hackers como el MIT. Dos fueron las innovaciones de Linus Torvalds: hacer extensivo el modelo bazar a escala planetaria a través de Internet y confiar a este modelo un proyecto de gran envergadura y complejidad como era el desarrollo de un kernel (que incluso para la Fundación para el Software Libre había sido una tarea pendiente por su dificultad). La explosión de Internet a principios de los noventa, integrando las redes existentes en la World Wide Web, fue condición necesaria para que este modelo pudiera proliferar hasta un punto tan alto.

La denominación modelo bazar fue creada por Eric Raymond (Raymond, 2009). Se contrapone al *modelo catedral*, en el que se basa el desarrollo de software propietario y que también seguía la Free Software Foundation en proyectos de alta complejidad. Por modelo catedral se entiende un modelo de desarrollo planificado y desarrollado de manera centralizada y jerárquica. Se creía que sin un planeamiento central y un cuidadoso trabajo de concepción previa, en el que cada etapa estuviera meticulosamente planeada y se ajustara a un cronograma, no se podía llevar a cabo un proyecto realmente complejo. También era necesario,

según esta visión, mantener grados de jerarquía y guardar celosamente los desarrollos hechos hasta el momento de liberar la primera versión, es decir, una vez que el producto estuviera terminado.

Ahora bien, en la práctica el desarrollo de software libre no pudo seguir este camino. Ya a Stallman se le hizo imposible crear GNU por completo para luego publicarlo y empezó a liberarlo por partes que se implementaron en los sistemas Unix del momento. Fue gracias a esto que ganó gran parte de las adhesiones a la Free Software Foundation. Incluso en los foros de Unix existía un antecedente del modelo bazar en las experiencias de feedback. Sin embargo, la cooperación de los consumidores en el desarrollo de un producto era algo muy subestimado y, hasta Linux, los proyectos más ambiciosos eran concebidos siguiendo el modelo catedral.

Linux vino a demostrar la verdadera potencia creadora de la cooperación libre entre cerebros que se da en el citado modelo. La ley principal del modelo bazar es liberar lo más rápido posible las nuevas versiones del software para que queden al alcance de toda la comunidad. De esta forma, cada uno puede retomar un trabajo de su interés, hacer las mejoras que según su criterio sean necesarias, hacer las sugerencias que desee y agregar nuevas características si le parece; luego el resto de la comunidad juzgará y retomará, para seguir reformulando, su nueva versión. A través de Internet, el acceso libre de los interesados al proyecto constituye un público a escala planetaria. Los consumidores del producto pueden aprender del código fuente e iniciarse en el software, los problemas pueden aislarse y solucionarse rápidamente. Ya que no hay instancias jerárquicas, cada individuo se autoorganiza según sus propios intereses. Este modelo, al decir de sus partidarios, es muy superior desde el punto de vista técnico, los avances se dan más rápido y no hay trabas burocráticas a la hora de resolver un problema.

El modelo bazar es el trabajo de la multiplicidad, de los muchos en tanto que muchos. En este contexto, no hay limitaciones a lo que se puede agregar o modificar, posibilitando que la creación y la innovación proliferen. Sin barreras jerárquicas claras y rígidas, solo los resultados cuentan a la hora de elegir qué nueva versión del producto es la que se incorporará a la nueva versión oficial. Sin embargo, no es necesario que se utilice la versión oficial; cualquier versión que parezca preferible al usuario es compatible con su entorno de software y hardware o puede ser compatibilizada.

Surge entonces la figura del prosumidor: un consumidor que, interesado por el producto que consume, aporta ideas, señala errores o los soluciona él mismo para luego compartir su creación, e incluso aporta las modificaciones que según él sean necesarias o le interese ver. O también, se puede hablar de prosumidores inconscientes, que con la sola utilización de sus programas están enviando valiosos datos a los desarrolladores. Con el código fuente libre, cualquier persona dispuesta a dedicar algo de su tiempo a aprender puede iniciarse en la programación aunque sea en un nivel muy básico. Para Nick Dyer-Witheford, en la relación entre productores y prosumidores “los equipos remunerados de los desarrolladores profesionales se convierten así en un mero núcleo de un conjunto difuso de creatividad que implica a creadores no remunerados, personas que se someten a los ensayos, informadores

expertos y trabajo voluntario" (Dyer Witheford, 2004: 54).

Según Raymond, la causa por la cual el software de la comunidad de software libre bajo el modelo bazar es tan alta es que las innovaciones y las mejoras al software parten de necesidades o intereses personales de los programadores que contribuyen. No solo esto enriquece el producto, sino que el programador trabaja con más entusiasmo y dedicación si realmente le interesa lo que hace. Podríamos decir que en cualquier actividad esto es así.

Marx, en uno de sus primeros escritos sobre economía redactados en el año 1844, hablaba de la alienación del trabajador con respecto a lo que produce (Marx, 1974). Con ello se refería a que lo que hace el obrero no le pertenece, lo siente ajeno a sí mismo, pero también al hecho de que, al trabajar por obligación, muchas veces lo hace sobre cosas que no le interesan ni le despiertan ningún entusiasmo. De esta manera, el trabajo se convierte en algo tedioso y pesado, ajeno, alienado. En el modelo bazar, en cambio, cada uno trabaja en lo que le entusiasma, y siente al programa como algo suyo, siente cómo una parte de sí mismo se incorpora en el software y el trabajo pasa a ser una actividad gratificante. Trabaja con pasión y dedicación en lo que ama, es decir, lo que todos soñamos que debería ser nuestra profesión.

Podríamos buscar más relaciones entre lo que venimos exponiendo sobre el modelo bazar y la obra de Marx, y ganarnos que nos miren raro exponiendo al modelo bazar como una posible solución al trabajo alienado. Este modelo no carece de problemas, pues existen muchos métodos de captura pro parte del capitalismo que, como luego veremos, restituyen la condición alienante al trabajo. Pero cuando eso pasa, aun cuando sigamos hablando de modelo bazar, no diremos que estamos ante trabajo libre, pues el trabajo alienado en el modelo bazar es completamente posible. Otra salvedad que se hace necesario hacer es que el trabajo puede seguir siendo libre sin dejar por ello de ser capturado por un flujo capitalista. Y, en última instancia, que a través de la modulación y direccionamiento por medio de diversos mecanismos publicitarios o similares se puede inducir a los prosumidores a que actúen con todo gusto y toda la gratificación mientras se les arranca trabajo no pagado, como efectivamente sucede en muchísimos casos en el mundo del software libre. Esto es fundamental para comprender qué tipo de negocio se hace con el software libre. Es decir, no hay que dejar de observar que el trabajo libre o no alienado también genera valor. Podría verse este trabajo como el fin de la fábrica o de la empresa en su forma clásica de organización fordista-taylorista, pero también existe una discusión que expone que en realidad lo que ha sucedido es que la sociedad entera se ha convertido en una fábrica, como lo exponen por ejemplo Maurizio Lazzarato y Antonio Negri (Lazzarato y Negri, 2007). Llegamos así a la conclusión de que, por positivos que sean el modelo bazar y el trabajo libre, tiene sus pros y sus contras, y que no existe aún sobre el mundo ningún paraíso.

En el párrafo anterior expusimos cómo en el modelo bazar se resolvería el problema de la alienación del trabajador con respecto a su producto y con respecto a la actividad productiva, al hacer algo que le interesa y le gusta y sentir cómo una parte de sí queda en el producto. La tercera categoría de alienación es del ser humano con respecto a su ser genérico, y acá el problema se hace más complejo. La alienación con respecto al ser genérico comporta dos

dimensiones: la dimensión natural (ver a la naturaleza simplemente como un medio y explotarla irresponsable e indiscriminadamente) y la dimensión social (ver al otro como un enemigo y un competidor). La dimensión social se resuelve fácilmente —al menos en la teoría—, pues en el modelo bazar la cooperación surge espontáneamente y los demás son colaboradores y colegas antes que competencia. En el modelo bazar, los otros no buscan competir conmigo sino cooperar en mi proyecto, y si alguien hace una versión mejor, es una ganancia para toda la comunidad y no para sí mismo. Y aunque sí existe la competencia, es, como diría Raymond, una competencia de egos, y la competencia así entendida no es como la competencia en el mercado, sino más bien como la que se da en los juegos, sana competencia, digamos, y aunque fuera insana (¡cuántos hackers se obsesionarán para hacer la mejor versión descuidando el sueño y su salud!), produce un bien que beneficia a toda la comunidad.

Con respecto a la dimensión natural, no podemos asegurar nada, pero tal vez el trabajo libre y no alienado genere una mayor interacción con el ambiente de producción, incluida la naturaleza. Pero parece inverificable, ya que nuestra investigación se circunscribe al software. Sin embargo, si se investigara la dinámica productiva en el movimiento de campesinos sin tierras es probable que se encuentre que algo así está efectivamente sucediendo. No estaríamos errados si diéramos por hecho que solucionar la alienación del ser humano con respecto a la naturaleza es un tema mucho más complejo que involucra muchos más factores, como la incorporación de muchos avances de las ciencias naturales al sentido común y cambios a nivel político y social. Digamos que se trata de otro mundo que requiere que se piensen nuevos mundos posibles.

Finalmente, no vamos a poder cumplir con la idea de presentar al modelo bazar como solución al trabajo alienado después de todo, pero al menos lo intentamos, y en el mundo del software libre siempre se tienen en alta estima los intentos y la iniciativa. La cuarta y última determinación del trabajo alienado es la del hombre con respecto al hombre. En este punto Marx se refiere a la explotación del obrero por parte del capitalista. Podríamos sostener que el copyleft, un instrumento jurídico diseñado por Stallman para evitar que el software libre sea transformado en propietario, es un arma contra esto, pero dado que existe la posibilidad de que una empresa capture la cooperación libre y se apropie de trabajo no pagado, la alienación persiste, como ya lo dijimos. Es, entonces, en esta determinación donde, como siempre lo expuso la teoría marxista, se dan los mayores conflictos. Las luchas contra las patentes y el copyright, y por el copyleft, que más adelante veremos, son un claro ejemplo de las tensiones que existen.

4. Dinámicas de cooperación y creación colectiva en el modelo bazar

En este apartado vamos a tratar de aproximarnos a las dinámicas de creación colectiva del modelo bazar desde las categorías de la filosofía de la multiplicidad. Ya avanzamos la primera de las categorías muy utilizadas por esta filosofía: el general intellect, concebida por Karl Marx y actualizada, entre otros, por Paolo Virno (Virno, 2003: 34-40). Utilizaremos también en este punto la noción de *agenciamiento,* introducida en la filosofía por Gilles Deleuze y Félix Guattari

y en boga actualmente en el lenguaje de diversos filósofos y teóricos. Entre ellos, destacamos a Maurizio Lazzarato. La descripción de la noción de agenciamiento en toda su complejidad escapa a los límites de este trabajo, pero a grandes rasgos se podría entender como un ordenamiento de componentes heterogéneos que en conjunto tienen una dirección, es decir, actúan en determinado sentido o para conseguir tal o cual fin. Pero un agenciamiento no es un todo con voluntad propia, sino que es una multiplicidad que persiste, en la que cada componente (sean personas, máquinas, herramientas, animales, etc.) mantiene su autonomía. Existen dos tipos de agenciamientos: un agenciamiento en el orden de lo simbólico, es decir, un agenciamiento de mentes, se denomina *agenciamiento expresivo* o *en el orden de la expresión*; un agenciamiento de cuerpos o de elementos materiales es llamado *agenciamiento maquínico*. Según Guattari, los componentes de los agenciamientos son heterogéneos y pueden ser "también de orden biológico, social, maquínico, gnoseológico" (Guattari, 2004: 133). Gracias a la noción de agenciamiento podemos entender la convergencia de distintas individualidades en proyectos comunes a pesar de la dispersión y bajo nivel de compromiso que Internet propicia, y podemos ofrecer una explicación a la conformación de grupos y tribus sin perder de vista que cada persona es un individuo único e irreductible. Pero también, gracias a la noción del general intellect como base común, evitamos caer en el error de concebir a la sociedad como una mera suma de individuos, ya que los individuos no crean a la sociedad por mera agregación, sino que la sociedad crea a los individuos a partir del general intellect.

Apliquemos estos conceptos a nuestro objeto de estudio. Decíamos anteriormente que el modelo bazar posibilita la cooperación libre entre cerebros. En esta cooperación libre, cada usuario puede desplegar su potencia de creación y coordinar con otros usuarios concibiendo posibles, virtualidades de diversa índole (desde proyectos de programas de software hasta manifestaciones de índole política), para luego efectuarlas a través de la dinámica de trabajo libre del modelo bazar que describimos en el apartado anterior. Los agenciamientos expresivos (es decir, agenciamientos simbólicos, como por ejemplo la creación de proyectos y alternativas antes de su efectuación) se constituyen a través de la red en la concepción de proyectos de software libre (como GNU o Linux) que, a la hora de consumarse en agenciamientos maquínicos (es decir, de llevarlos a cabo en un producto concreto), agencian también la cooperación de los consumidores del proyecto, que en el modelo bazar pasan a ser cooperadores en las sucesivas mejoras y nuevas versiones del software, esto es, en prosumidores. Esto queda patente en cómo cada usuario de Linux puede cooperar aportando desde nuevas versiones hasta sugerencias e ideas. En la cooperación entre cerebros hay entonces una actividad de cocreación y correalización entre las personas que integran el público del software libre (en este caso), quienes intervienen expresando diversas potencias y grados de creación y agenciamiento.

Para existir, la cooperación libre que describimos depende de que los individuos puedan acceder a bienes comunes que resultan del desarrollo y la difusión de la ciencia, es decir, al intelecto general, a Internet y a las redes de comunicación y conocimiento, además de poseer capacidad de operar los dispositivos tecnológicos. Con una base general de tecnología y

conocimiento, cualquier persona puede ingresar al público y efectuar su potencia de creación. En otras palabras, cualquiera que tenga acceso a Internet, una computadora medianamente aceptable y el conocimiento básico (mínimo) de cómo operarla (y sepa algo de inglés) puede, si realmente siente interés, ir entrando de a poco en la comunidad del software libre e iniciarse a cooperar con ideas, opiniones, escribiendo manuales, documentación, etcétera. Con perseverancia y entusiasmo podrá ir aprendiendo de a poco sobre programación. Para decirlo en palabras del comunicólogo Antonio Pasquali, el acceso a los medios para formar parte de la sociedad no es suficiente, pero es una condición indispensable para la participación (Pasquali, 2007: 72-78).

La invención de software, por lo tanto, se basa en esta cooperación entre individuos, este agenciamiento de una multiplicidad de inteligencias que dependen de una base genérica común para converger en un público determinado, independientemente de la clase social a la que pertenecen. No obstante, no olvidemos que muchísimas veces existe un determinismo económico que impide el acceso a los medios tecnológicos y cognitivos necesarios, pero eso queda afuera de los límites de este trabajo.

Esta multiplicidad en principio está constituida por los creadores de un software (en nuestro caso, aunque podría hacerse un análisis semejante para otros bienes o incluso movimientos sociales), pero a medida que se difunde, la multiplicidad de los usuarios es agenciada por los creadores y nos encontramos ante la convergencia entre consumidor y productor de la que ya hemos hablado (prosumidor).

Sin embargo, no todo es prodigio en esta dinámica. En la sociedad actual, muchas veces esta cooperación es capturada por el capitalismo. Esto no es nada exclusivo del capitalismo cognitivo: a lo largo de su historia, el modo de producción capitalista ha necesitado siempre apropiarse de elementos externos a la producción en sentido estricto para que esta funcione. Es el caso, por ejemplo, de la división sexual de roles. Para funcionar, el capitalismo necesitaba que los obreros recuperaran la fuerza perdida (y la autoestima, por supuesto, que muchas veces queda maltrecha luego de arduo un día laboral). A esta tarea estaban destinadas las mujeres, quienes se encargaban (realizando la función que era considerada propia de la mujer) de las tareas del hogar y de reproducción de la fuerza del trabajo. Esas tareas, necesarias para el proceso de producción, eran trabajo no pagado, pues a través de todo un proceso de desvalorización del trabajo doméstico (necesario para que se justificara su condición de trabajo no pago), se le quitó el estatus de trabajo propiamente dicho y, aunque esto en la práctica no sea cierto, se impuso la idea de que el trabajo doméstico es un trabajo menor, que supone poco esfuerzo y que no contribuye a la creación de valor. Además, también a través de la desvalorización de la mujer, el hombre podía mitigar la frustración que traía del trabajo descargándola en ella. Pero esto ya es otra historia que daría lugar a otros trabajos. El punto era presentar un ejemplo clásico de cómo el capitalismo arranca trabajo no pagado expulsándolo fuera de la esfera de la valorización capitalista.

Hoy sucede algo similar cuando se captura la cooperación libre a través del software. Ahora bien, en la unión de consumidor y productor (prosumidor) que se da en el modelo bazar,

capturar la cooperación supone capturar la clientela. De esta forma, esta captura opera de tal manera sobre la cooperación que pareciera que la capacidad de creación queda completamente del lado de la empresa, transformando a los cooperadores en meros consumidores. Sin embargo, aún se les arranca trabajo no pago de diversas formas. Por ejemplo, cuando nos quejamos por un error, o cuando habiendo fallado un programa aparece un aviso que dice “enviar informe de errores”, esto supone un trabajo de aislamiento e identificación de un bug (= error de programación) que nadie nos reconoce. Bien podría alguien objetar que “eso no es arrancar trabajo libre, eso es feedback y mantenimiento”, cosa que es verdad, pero es necesario tener en cuenta que muchas veces estamos sobre versiones “beta” (inconclusas) de programas o estamos cooperando, sin saberlo, en el proceso de creación de la nueva versión del software. Incluso muchas veces el programa nos propone: “ayúdanos a mejorar nuestro software”.

Se produce así una captura unilateral por parte de la empresa, y ya no podemos hablar de cooperación libre entre cerebros sin señalar los límites de esta afirmación, pues las empresas privadas modulan y direccionan la cooperación en su beneficio. Los productos de software propietario ocultan su código fuente, por lo tanto en la captura la potencia de creación y agenciamiento ya no está distribuida equitativamente: la empresa es la instancia dominante. El copyright y demás leyes sobre propiedad intelectual son los mecanismos para producir la captura unilateral. En este marco, el copyleft es un mecanismo de resistencia en tanto que evita que el código producido como software libre sea privatizado. Sin embargo, para la empresa lo que importa no es apropiarse del producto en sí, sino de la base de usuarios que captura junto al programa y, si bien existe el copyleft, hay muchas empresas que buscan crear alianzas con productores de software libre para capturar su base de usuarios y también hay movimientos dentro del software libre que utilizan licencias que permiten que su código sea privatizado (Lazzarato, 2004).

A través de los mecanismos de captura de las multiplicidades para agenciarlas como clientela y de la subsiguiente modulación de su capacidad creativa, la empresa no anula la capacidad creativa de los usuarios, sino que la encausa en su propio beneficio, operando una captura unilateral. Así, se reduce la actividad de creación colectiva a una actividad cognitiva para modular la cooperación.

5. Conclusión

El movimiento social del software libre se encuentra en el medio de una serie de encrucijadas frente a los constantes cambios que sufre el mundo de la informática y los incesantes embates e intentos de captura de las lógicas de acumulación capitalistas. Al ser un movimiento disidente cuya lucha se desenvuelve en uno de los núcleos más importantes de la producción e innovación actuales, se encuentra en constante turbulencia y tensión con sus detractores. El software libre cuenta con un importante logro: el haber desplegado un importante modelo de cooperación y trabajo libre, el modelo bazar, que se contrapone con la racionalidad dominante del capitalismo imperante.

No obstante, al final de nuestro trabajo surge una duda que ya se podía vislumbrar desde párrafos anteriores: ¿es el modelo bazar una verdadera solución y un punto de partida para nuevos modos de producción y acumulación o se trata solo del preludio de un nuevo tipo de capitalismo aún más total que pueda agenciar a su maquinaria todas las disposiciones del intelecto general humano? Varios indicios parecen señalar como muy posible la segunda opción. Sin embargo, el modelo bazar tiene muchas ventajas frente al todavía imperante, aunque oxidado, viejo modelo de producción capitalista de los siglos pasados. Es necesario que quienes defendemos la cultura libre no bajemos la guardia y estemos preparados para rescatar al modelo bazar como un espacio para la cooperación de la multiplicidad y que este no sea apropiado por el capitalismo, para así evitar que, una vez más, la libertad no sea definida más que como libertad de comercio.

Notas

1. Nuestra fuente es el trabajo "Breve historia de la cultura hacker," del ensayista y hacker Eric Raymond (en Gradín, C. [2004], *Internet, hackers y software libre.* Capítulo 2. Buenos Aires: Editorial Fantasma).
2. Código fuente es el código sobre el cual se desarrolla, perfecciona, modifica o repara un programa informático.
3. En este apartado trabajaremos a partir de las ideas de Stallman, R. (2004). *Software libre para una sociedad libre.* Madrid: Traficantes de sueños y Vidal, Miquel en Gradín, C. (2004). *Internet, hackers y software libre.* Buenos Aires: Editorial Fantasma. Capítulo 3: "Cooperación sin mando: una introducción al software libre".

Bibliografía

Dyer-Whiteford, N. (2004). "Sobre la contestación al capitalismo cognitivo. Composición de clase de la industria de los videojuegos y de los juegos de ordenador". En Blondeau, O.; Dyer Whiteford, N.; Vercellone, C.; Kyrou, A.; Corsani, A.; Rullani, E.; Moulier Boutang, Y.; Lazzarato, M. (2004). *Capitalismo Cognitivo, propiedad intelectual y creación colectiva.* Madrid: Traficantes de sueños.

Gradín, C. (2004). *Internet, hackers y software libre.* Buenos Aires: Editorial Fantasma.

Guattari, F. (2004). *Plan sobre el planeta.* Madrid: Traficantes de sueños.

Lazzarato, M. (2004). *Por una política menor. Políticas del acontecimiento en las sociedades de control.* Madrid: Traficantes de sueños.

Lazzarato, A. y Negri, A. (2007). "Trabajo inmaterial y subjetividad". En *Revista Brumaria 7 - Arte, máquinas, trabajo inmaterial* [en línea]. Dirección URL: http://www.brumaria.net/publicacionbru7.htm [Consulta: 19 de enero de 2011].

Marx, K. (1972). *Elementos fundamentales para la crítica de la economía política (Grundrisse) 1857-1858,* vol. 2. México: Siglo XXI.

Marx, K. (1974). *Manuscritos: economía y filosofía. Primer manuscrito.* Madrid: Alianza editorial.

Merton, R. (1977). *La sociología de la ciencia,* vol. 2. Madrid: Alianza Editorial.

Pasquali, A. (2007). *Comprender la comunicación.* Barcelona: Gedisa.

Raymond, E. (2009). *La Catedral y el bazar.* Buenos Aires: Openbiz.

Virno, P. (2003). *Gramática de la multitud. Para un análisis de las formas de vida contemporáneas.* Madrid: Traficantes de sueños.