Notas para un estudio de los trabajadores informáticos.

Marcelo Raimundo (UNLP)

Desde mediados de los años ’70 el capitalismo ha sufrido una serie de mutaciones con respecto a la dinámica que había adoptado desde la posguerra. Dichas transformaciones han llevado, entre otras cuestiones, a una reformulación en la manera de caracterizar sus dos polos básicos: el capital y el trabajo. Por un lado, la crisis del fordismo como modo de regulación y la disminución del volumen de mano de obra ocupada en el sector industrial, llevaron al desarrollo de la hipótesis del “fin del trabajo”. Por otro, la revolución microelectrónica e informática acompañada de un fuerte proceso de valorización del trabajo intelectual, se entendió como la emergencia de un capitalismo “posfordista”, “informacional”, “cognitivo”, etc. Más allá de las discusiones en torno a categorizaciones o lógicas dominantes, lo que nos interesa aquí es tomar nota de la emergencia de un nuevo colectivo de trabajadores que está íntimamente relacionado con la nueva situación, los vinculados a las *TIC’s* (tecnologías de la información y la comunicación).

El objetivo de esta ponencia es realizar una primera aproximación a ciertas cuestiones y elementos a tener en cuenta para encarar una investigación social e histórica sobre este nuevo tipo de trabajador, que caracteriza hoy a gran parte de los sectores productivos de punta. Para ello nos centraremos en los *trabajadores informáticos*, es decir a los que se dedican a la producción de *software*. El interés en este sector particular, proviene de que en

los últimos años ha manifestado en su seno la emergencia de ciertos fenómenos que incorporan tanto formas como contenidos que podrían considerarse novedosos en relación a la expresión del conflicto de clases en general.

I

En un reciente artículo orientado al análisis de los procesos de trabajo, Beirne, Ramsay y Panteli (1998) afirman que la literatura al respecto se ha interesado por la computación más por los efectos que produce sobre el trabajo de otros, que por analizar el trabajo que hay detrás de ella. De esta manera, el trabajo que produce *software* (es decir los programas con los cuáles funcionan las computadoras y robots) ha quedado como una "caja negra". Sin embargo, dichos autores señalan que la importancia de estudiar este trabajo específico estaría dada al menos por tres cuestiones: 1) por su crecimiento como categoría laboral mientras otras desaparecen, incrementando su significación para el futuro, 2) por ser una categoría ocupacional que no arrastra una larga historia, 3) por que los resultados de este proceso de trabajo tiene directas y poderosas implicancias sobre el futuro del trabajo en general.

El trabajo informático como forma de trabajo concreto nace a principios de la década de los años cuarenta en el marco de la segunda guerra mundial y vinculado fundamentalmente a proyectos desarrolladas por las fuerzas armadas estadounidenses. Los científicos e investigadores abocados a la creación la primer computadora, la *ENIAC (*Electronic Numerical Integrator And Computer), asumieron por igual las nuevas tareas que esta

empresa demandó: el diseño y la construcción del *hardware* (lo físico-material) y del *software* (lo lógico-inmaterial).

Al calor del desarrollo de la computación estas tareas comunes se irían haciendo específicas, dando como resultado una rama particular que se encargará especialmente de la producción de software, que aparece claramente conformada recién en los años '60. Quizás algunas de las singularidades más notorias de esta ocupación en sus inicios hayan sido: a) como señala Barrett (2001), en la programación temprana no había una clara división del trabajo, una misma persona se encargaba de realizar varias tareas dentro del mismo proceso productivo, por ejemplo del diseño, entrada de datos, depuración, mantenimiento, etc.; b) en cuanto a la calidad de la tarea efectuada, Beirne, Ramsay y Panteli (1998) afirman que el desarrollo de software fue considerado como una tarea "creativa y esotérica" y con altos grados de autonomía, y la computación se planteó más como un "arte" que como una ciencia, cuestión que aún hoy encuentra eco en términos como *digital artisan* y que más que retratar una realidad extendida es una expresión de deseos de la mayoría de los trabajadores de este sector, como veremos más adelante.

Si bien podemos afirmar que la computación tuvo un nacimiento cobijado por el estado, rápidamente fue incorporada al mundo de los negocios y sobre todo -en esta primer etapa- el procesamiento de datos se canalizó al sector servicios, siendo el ejemplo más emblemático la empresa multinacional *IBM* (International Business Machine). Durante este período temprano, la producción del hardware y del software se consideraban parte de un mismo proceso, ya que las empresas informáticas vendían sus computadoras incluyendo el software necesario para su funcionamiento, ambos se consideraban un único producto.

Cuando se advirtió que el software podía ser valorizado y transformado en una mercancía más, independiente de su soporte físico, los senderos se bifurcaron.

Las consecuencias de esto no tardaron en aparecer y para fines de los años sesenta se empieza a hablar por primera vez de una "crisis del software". Allí comienza el proceso de transformar este *black art* en una ciencia, la ingeniería de software, un esfuerzo racionalizador que continúa hasta la actualidad, ya que no ha encontrado pocas resistencias. En referencia a esto, tendríamos que diferenciar dentro de la continuidad que comprende el proceso de racionalización de la producción de software al menos dos etapas. Por un lado, el período al que nos estamos refiriendo que llega hasta fines de los años '70, caracterizado por un marco tecnológico asociado a las grandes computadoras, las *main-frames*. Por otro, la etapa que se inicia con la llamada "revolución microelectrónica", que implicó un salto cualitativo en la producción de software. Según Katz (1996) en 1977 existían 40.000 computadoras en todo el mundo, veinte años después se fabricaba ese número diariamente. El lanzamiento de la PC (*personal computer*) produjo la difusión a nivel general de la informática, su presencia tanto en los más variados lugares de trabajo, desde las oficinas hasta las fábricas, como en la mayoría de los hogares urbanos, podríamos decir una producción y consumo de masas. Es en este marco en el que se dan las discusiones más recientes sobre la transformación de los procesos laborales dentro del sector, que revelan aspectos particulares del proceso de racionalización del trabajo informático.

En este sentido varios autores, entre ellos Greenbaum (1995) y Orlikowsky (1988), propusieron la tesis de la *proletarización* de los trabajadores del software, asociada a un escenario de descalificación y degradación, en clara consonancia con las tesis de

Braverman (1987). Beirne, Ramsay y Panteli (1998) aseguran que de hecho se revelan una serie de nuevos elementos racionalizadores: a) la automatización de la programación, vinculada a los nuevos lenguajes de 4° generación, la programación 'orientada a objetos', los lenguajes *visual* y herramientas tipo *CASE*; b) la 'industrialización' de la producción de software, ligada a la crecimiento de grandes establecimientos (*large software houses*, *software factory*), en detrimento de la producción *in-house*, es decir dentro de las mismas compañías que requieren los programas, profundizando la rutinización e intensificación en los procesos de trabajo; c) la ampliación del control en los procesos (*bureaucratic controls*), que ya no se reducen sólo a la implantación de una disciplina como la ingeniería de software, sino que se relacionan con las actuales políticas de productividad y calidad, como por ejemplo, la aplicación de normas como las series ISO o las CMM; d) la mercantilización (*commodification*), donde la producción se abre a las presiones del mercado, para aumentar la eficiencia, cortar costos e imponer una autodisciplina en los trabajadores, siendo un claro ejemplo de esto la introducción del cliente (*end-user*) en el proceso productivo.

La línea bravermaniana de análisis fue hegemónica en los primeros estudios críticos sobre el trabajo informático, y buscó comprobar que la tendencia a la *taylorización* de los procesos laborales bajo el capitalismo se verificaba también en este sector productivo ligado fuertemente al trabajo intelectual. Un ejemplo de ello son las conclusiones de los conocidos especialistas en el tema Kraft y Dubnoff (1986), que han señalan que si bien la programación originalmente había sido un coto de artesanos que respondían principalmente a sus propias visiones, hacía los años setenta esto fue en gran parte eliminado, transformado

a la programación de computadoras en una rama de la ingeniería, con normas de producción definidas y dirigidas, y escalas de pago.

Entrada la década de los noventa, comienzan a aparecer una serie de cuestionamientos a dicho "absolutismo bravermaniano", que en base a una serie de estudios empíricos en empresas dedicadas a la producción de software, implicaron no sólo una revisión de las transformaciones sufridas por los procesos laborales en esta rama de la producción, sino que también reformulan las imágenes totalizadoras que se derivaron de la caracterización de lo que hoy se da en llamar *posfordismo* o *toyotismo*. Podemos ver planteada esta cuestión en Beirne, Ramsay y Panteli (1998), que por un lado afirman que en lugar de cumplir una lógica ineluctable, estos cambios muestran alguna fluidez, pero a la vez reconocen los límites de la implantación de las nuevas estrategias, pues a pesar de la imagen tradicional de equipos de software que refuerzan los lazos de grupo y permiten expresar autonomía, los desarrolladores están experimentando una fragmentación de los equipos de los proyectos y un funcionamiento más individualizado. En síntesis, desafiar una simple tesis de descalificación/fragmentación no debería conducir a exagerar la influencia que tienen los trabajadores informáticos sobre el proceso de desarrollo, es decir la tan promocionada autonomía de los nuevos paradigmas productivos.

A la hora de establecer posibles explicaciones a estas "tendencias contradictorias" en la evolución del trabajo informático, encontramos al menos dos tipos de estrategias de análisis. Una es la desarrollada por Barrett (2001), que se vincula al tipo de producto fabricado, diferenciando entre el software primario, que comprende al software *empaquetado* o *standarizado*, por ejemplo sistemas operativos, lenguajes, procesadores de

texto, etc.; y el software secundario, que se desarrolla en base al anterior y refiere a los sistemas realizados *a medida*, también conocidos como servicios informáticos. Dicha autora tiene la hipótesis que la existencia de una jerarquía de productos de software implica una diferencia en los procesos de la producción y estrategias para controlar el proceso de trabajo, que puede vincularse a los dos paradigmas de desarrollo de software conocidos como *formalista* y *pragmático*. El llamado formalista es el que sigue la línea de la anteriormente mencionada ingeniería del software -cuyo modelo clásico es la *waterfall* (cascada)- y de la programación estructurada, y está asociado en general a la producción de software secundario. El paradigma pragmático, ligado a la fabricación de software primario, se estructura en base a *teams* o grupos de desarrolladores que gozan de un alto grado de auto-dirección y libertad, ya que pueden elegir sus herramientas de producción y su organización interna, y donde los *managers* actúan como mentores dedicados más a "inspirar que a azotar". Sin embargo, esta autora no deja de reconocer que la disciplina no desaparece, sino que ya no se ancla en las funciones tradicionales de los *managers*, sino que se rinden cuentas de la productividad por otros mecanismos, entre ellos el más conocido es el *daily build*, regido por la óptica de la calidad total y la presión comercial. Barrett también señala, como para sumar complejidad al asunto, que ambas estrategias de control pueden funcionar simultáneamente en una misma empresa, dando como resultado una estrategia "híbrida".

Otra de las formas de interpretar la ambivalencia de los procesos racionalizadores, es tener en cuenta la resistencia que le oponen los trabajadores. En relación a esto podemos tomar nota de varias cuestiones que señalan Beirne, Ramsay y Pantelli (1998): a) se reconocen desviaciones de las metodologías oficiales de desarrollo, donde las llamadas 'buenas

prácticas' son reemplazadas por prácticas *ad hoc* y "caóticas"; b) se admite que las prácticas de calidad no penetran los niveles bajos de desarrollo; c) se advierte un rechazo o limitado uso de nuevas tecnologías, por ejemplo de los lenguajes de 4° generación y de las herramientas *CASE*; d) se observa que los *managers* se ven obligados a negociar la estrategia de desarrollo teniendo en cuenta las llamadas *tacit skills* e *informal practices*; e) se perciben resistencias a la incidencia de los *end-users* en el proceso productivo. Si bien la raíz de la resistencia a la racionalización se deberían según Kraft y Dubnoff (1986) a los conflictos entre dirigentes y dirigidos en torno al contenido de las tareas, a la definición de calificaciones y en general al control del trabajo, Beirne, Ramsay y Pantelli (1998) reconocen que muchas de estas actitudes derivan de la contradicción que surge del intento de aplicar simultáneamente una lógica de control según los modelos ingeniería de software, con una estrategia posfordista de presión de mercado y autonomía de los grupos.

En realidad, estos señalamientos resultan atractivos a la hora de analizar este colectivo laboral ya que siempre se ha reconocido en estos trabajadores un alto grado de compromiso en relación a su trabajo, de ahí la imagen generalizada del informático como un *nerd*, persona apasionada con su tarea, que pasa horas sentado frente a una computadora hasta resolver el trabajo asignado. Incluso Beirne, Ramsay y Pantelli (1998) reconocen que muchas de las resistencias a las políticas de *management* no van en detrimento de la productividad, sino al contrario, los trabajadores toman la iniciativa y desarrollan recursos propios, métodos *ad hoc* y actividades informales con el objetivo de mejorar el software que producen.

Como síntesis de los aspectos hasta aquí tratados, podemos señalar que al hacer este recorrido sobre literatura que aborda el proceso de trabajo informático se revela una tendencia que sutilmente va progresivamente desdibujando la incidencia del componente de *clase* en el análisis, que si bien nunca lo encontramos definido precisamente, estaba presente en la base de los estudios en clave bravermaniana. Incluso si observamos los estudios que abordan los conflictos desde el punto de vista de la identidad, -por ejemplo Lockyer, Marks y Mulvey (2001a) y (2001b)-, si bien se reconoce el peso de la flexibilidad laboral y del ascenso del *outsourcing*, las identidades que entran en conflicto son las de tipo profesional, las grupales y las organizacionales o empresariales.

Ahora bien, la paulatina desaparición de la dimensión clasista en el análisis de estos trabajadores no deja de sorprender, sobre todo si tenemos en cuenta los fenómenos que emergieron en los últimos años, en relación tanto a comportamientos y prácticas como a formas organizacionales y de conciencia dentro de este colectivo laboral. A continuación trataremos de exponer cuáles son estos fenómenos novedosos, que difícilmente podrían ser abordados profundamente sin tener en cuenta las relaciones de producción en la que se encuentran nuestros sujetos de análisis.

II

Si hay algo que no deja dudas a los investigadores del mundo del trabajo, es la llamada "crisis del sindicalismo", que se manifiesta desde mediados de los años setenta a nivel mundial. Quizás sus indicadores más indiscutibles sean la tendencia a la disminución de la

tasa de sindicalización, la caída de la conflictividad laboral y la pérdida de peso político de los sindicatos.

Si bien las razones esgrimidas de forma general en torno a temas como la automatización de los procesos productivos a raíz de la revolución tecnológica, el crecimiento estructural de la desocupación, el retroceso de la producción industrial y otras cuestiones que aluden en última instancia a la caída de la población ocupada -por ejemplo los trabajos de Gorz (1982) y Rifkin (1996)-, estos son más bien argumentos que asocian al trabajador con un tipo específico de obrero, el industrial, y han sido la base de la tesis del "fin del trabajo", tan de moda en la actualidad. Aunque asociar exclusivamente estos fenómenos con la crisis del sindicalismo puede ser fácilmente rebatible, pues como señala de la Garza Toledo (2001) "(e)l supuesto obrero de la industria pesada, calificado, hombre, con trabajo estable y sindicalizado, siempre fue una minoría en todas las épocas, y no en todos los países y épocas ha sido la vanguardia del movimiento proletario", no por ello habría que descartarlos de lleno a la hora de buscar explicaciones, sino más bien tratar de articularlos como lo hace Antunes (1999) con un proceso paralelo que él denomina *subproletarización*, que remite a la expansión del trabajo parcial, flexible, precario, tercerizado, vinculado a la economía informal, subcontratación, etc. Dicho autor afirma que todo este conjunto de cambios llevó a una metamorfosis tanto del aspecto material como subjetivo del trabajo, afectando entonces su *forma de ser*, que sería lo que está en la base de la crisis del sindicalismo. Pero aquí no habría que olvidar, que esta metamorfosis del trabajo no fue un proceso "natural", sino el resultado de una lucha a nivel mundial en la que una profunda ofensiva del capital se combinó con una decidida acción estatal, dando como resultado una

derrota de la clase trabajadora que arrastró al sindicalismo a la situación en que hoy se encuentra.

Sin embargo y paradójicamente, el sector informático y en general los llamados *IT workers* (trabajadores de las tecnologías de la información), han ido revelando una contratendencia. Aunque los trabajadores de empresas de alta tecnología han mostrado en general una gran prescindencia respecto a los sindicatos, en los últimos años han emergido una serie de emprendimientos de sindicalización a nivel mundial que se han revelado bastante exitosos. Esta tendencia es parte de una profunda reformulación que se ha planteado en amplios sectores del movimiento sindical a nivel mundial, básicamente en torno a los desafíos planteados por lo que han dado en llamar la *nueva economía*.

Esta nueva estrategia revela dos aspectos importantes. Por un lado, intenta asumir los desafíos que provienen de las nuevas formas de trabajo, como afirma un informe del 1° Congreso de la *Union Network International* (2001), "Una palabra clave en las estrategias de sindicalización debe ser la 'diversidad'. Los sindicatos nunca serán representativos si no velan por cubrir una variedad de trabajadores pertenecientes a todas las capas de la sociedad, incluidos las mujeres, los jóvenes y las personas que trabajan a tiempo parcial, no permanente y en modelos de trabajo diferentes, v.g. el teletrabajo". En este sentido resulta revelador de las nuevas estrategias emprendidas el comentario realizado por Mike Blain -presidente de *WashTech*, sindicato de trabajadores de la información del estado de Washington, donde se encuentra la casa matriz de *Microsoft*-, citado por Levinson (2001): "Nosotros tenemos que reinventar lo que significa ser un sindicato. Un sindicato que enfoca las cosas como la antigüedad y sueldos sin escuchar lo que los trabajadores IT quieren va a

tener dificultad para organizarlos. Muchos sindicatos o secciones no entienden que los problemas prácticos tradicionales necesariamente no están en la vanguardia de la mente de la mayoría de los trabajadores IT. Los problemas que están en sus mentes son el aprendizaje, el establecimiento de las normas para el desarrollo del software, la protección de sus beneficios, las horas extraordinarias forzadas y la visa H1-B". Salvando las distancias, podemos encontrar aquí ciertas similitudes con rasgos del espíritu sindical de los años 60 y 70, la preocupación por las *condiciones de trabajo*, que fue la llave de los movimientos clasistas y antiburocráticos de dicha etapa.

Por otra parte, se revela una importante incorporación de las nuevas tecnologías a la actividad sindical. Para el caso que nos ocupa esto ha sido algo esencial. Los sindicatos que hemos podido observar -*WashTech* (EEUU), *Alliance@IBM* (EEUU), *IT Workers Alliance* (Australia)-, se han organizado básicamente a partir del uso de redes informáticas, principalmente la *Internet*, es decir aprovechando los mismos instrumentos de trabajo, y han revelado un rápido crecimiento en los últimos años. Esta nueva forma de práctica organizacional es lo que Freeman y Rogers (2002) han dado en llamar *open source unionism*: "(m)ientras el sindicato tradicional está basado en un patrón en un lugar particular, un sindicato *open source* puede ganar sus miembros de muchos sitios e incluso muchos patrones, vía los lugares de trabajo, la *Web*, y las comunidades *on line*. Los sindicatos tradicionales pueden usar tácticas de huelgas, arbitraje, trabajo a reglamento y otros métodos, el sindicalismo *open source* puede confiar en métodos amplios, como el *cyber picketing*, campañas en los medios de comunicación nacionales, la presión de la comunidad, y así sucesivamente". Un recorrido por los sitios *web* de estos sindicatos aporta

una valiosa información acerca de las actividades desarrolladas y también retrata en qué consiste esta nueva forma de sindicalización.

En el caso de *Alliance@IBM* (que como su nombre lo indica es un sindicato de empresa, y que organiza un colectivo que excede a los trabajadores específicamente informáticos) se puede observar que, como antes señalábamos, hay un claro énfasis en el mejoramiento de las condiciones de trabajo, por ejemplo las *overtime policies* (sobretrabajo sin pago), además de abordarse temas como los planes de retiro y pensiones, la obtención de negociaciones colectivas, la garantía de derechos como la no discriminación por edad, la posibilidad de organización, etc.

IT Workers Alliance (un sindicato que agrupa en un sentido profesional), proclama sus objetivos en un *IT Workers Manifesto*: "1. $20 de salario mínimo por hora; 2. paga justa basada en claros niveles de habilidad; 3. 38 horas por semana -horas extras o tiempo libre en lugar de convenios con horas adicionales; 4. instrucción en el trabajo; 5. oportunidades de promoción basadas en entrenamiento y habilidades; 6. cinco semanas de vacaciones pagas y nueve semanas pagas por licencia de maternidad; 7. protección de la privacidad en el trabajo; 8. la portabilidad de derechos de trabajo al trabajo; 9. protección de derechos durante las tomas de control y colapsos; 10. reconocimiento de los derechos del trabajador sobre su trabajo creativo". Como se puede ver, hay mucha coincidencia en los reclamos, pero también aparece enfáticamente algo que resulta bastante novedoso como función o servicio sindical, el *training*, el entrenamiento y adquisición de nuevas calificaciones y habilidades, que en el nuevo paradigma productivo resulta fundamental para permanecer en el mercado de trabajo actual.

WashTech es un sindicato fundado en 1998 y miembro de la *Communications Workers of America* (CWA). Si bien en la actualidad agrupa a trabajadores de la industria *high-tech* en sus diversas variantes (empleados permanentes, contratados por agencia y contratados individualmente), nació en base a la lucha de los denominados *permatemps* (empleados temporarios de largo plazo) y en un marco de fuerte oposición patronal (principalmente de *Microsoft*), por lo que además de compartir los reclamos de los sindicatos antes mencionados (negociación de contratos, compensación por despido, condiciones de seguridad, ningún tipo de discriminación, licencia por enfermedad), pone un fuerte énfasis en el derecho a organizarse sindicalmente.

En general los sitios web de estas organizaciones procuran una gran cantidad de información para sus afiliados, que además de ser un tema bastante novedoso en cuanto a la democratización de las mismas, aporta datos sustanciales para la investigación: informaciones sobre empresas -tanto cuantitativas como de políticas aplicadas-, acciones laborales y sindicales, artículos sobre las distintas problemáticas que enfrentan los trabajadores, información legislativa y judicial, copias de los reclamos realizados tanto a la patronal como al estado, informes sobre otras organizaciones laborales, noticias de la industria en general, *links* (vínculos), cursos de *training*, etc. Quizás una de las informaciones más jugosas que brindan estos sitios sean las que aportan los propios afiliados, ya que una de las características de este nuevo sindicalismo es la comunicación *on-line* entre sus integrantes. En estas *webs* podemos hallar historias personales, informes realizados por los miembros sobre sus casos personales, los *help-desk*, y la existencia de los

llamados *forums* o *flames*, donde se producen interesantes debates en torno a varios temas con la participación de los miembros y activistas.

III

El otro fenómeno que ha marcado el desarrollo y la evolución de los trabajadores informáticos en tanto productores de software ha sido el denominado *software libre*, conocido a nivel mundial como *free software* y rebautizado en los últimos años con el nombre de *open source* (código abierto o fuente abierta). Su impacto a nivel global no sólo se reduce al aspecto económico, sino que se ha extendido también al ámbito político e ideológico, cuestión que ha derivado en los últimos años en un creciente interés por parte de diversos cientistas sociales por explicar su funcionamiento, analizar sus particularidades y establecer sus potencialidades de cara al futuro del trabajo y la producción.

Decíamos antes, que a nuestro parecer resulta imprescindible al abordar la emergencia e importancia de este fenómeno, tener en cuenta la dimensión de clase que atraviesa al mismo. Ya que esta afirmación es más que un supuesto una hipótesis de trabajo, expondremos aquí en términos muy generales algunas de las caracterizaciones, abordajes y discusiones que se han derivado del análisis por parte de las ciencias sociales del software libre, con el fin de resaltar la ausencia de las relaciones de producción como clave de análisis, paradójicamente, en un movimiento que lo que cuestiona fundamentalmente es la forma central de propiedad en este sector de la producción, la *propiedad intelectual*, forma de propiedad que muchos autores consideran central en la actual etapa capitalista, donde el

eje de la acumulación en importantes sectores de punta del capital está en la valorización del trabajo intelectual, en la forma de conocimiento, comunicación e información.

Sólo breves y superficiales referencias hemos encontrado en la bibliografía recorrida acerca de nuestro punto de vista. Por ejemplo, Hannemyr (1999) señala que "La emergencia de los *hackers* como un grupo identificable coincide estrechamente en el tiempo con la introducción de varios métodos tayloristas en el desarrollo de software. Muchos de los programadores más experimentados notaron lo que estaba pasando a su oficio. Una de las cosas que caracterizaron a los tempranos *hackers*, era su rechazo general a los principios y prácticas tayloristas, y su continuada insistencia en que el trabajo de computación era un arte y una destreza y esa calidad y excelencia en el trabajo de computación ha estado arraigada en la expresión artística y la artesanía y no en las regulaciones". Aquí cabe hacer dos aclaraciones. Por un lado, el término *hacker* está indisolublemente ligado a la emergencia del software libre, y no debe entendérselo con el sentido que vulgarmente tiene en la actualidad (en general, el de "pirata" informático, con claras connotaciones de ilegalidad), sino que el *hacking* esta haciendo referencia a una actitud y un método para la construcción de sistemas de información y artefactos de software, y en palabras de la misma autora, su origen se da como una reacción a los intentos de imponer un "modo industrial de producción" en el desarrollo de software. Por otro, si bien la temprana taylorización de la informática provocó esta respuesta, y esto nos indica la importancia del factor histórico a la hora de abordar el software libre, no deberíamos dejar de tener en cuenta que el mismo se ha desarrollado exponencialmente recién en los últimos años, lo que nos estaría indicando hipotéticamente que la experiencia temprana se encuentra de

alguna manera recreada continuamente, lo que estaría señalando quizás cierta particularidad de los que desarrollan este tipo de *trabajo concreto*.

Otra escueta alusión que abona nuestra línea de observación, la encontramos en Kogut y Metiu (2001) que afirman que "Es fundamental para entender el origen del *open source* reconocer la profunda hostilidad de los programadores a la privatización del software. El software temprano (...) fue creado en ambientes *open source* y libremente diseminado" y vinculan este proceso de "privatización" a la exclusión de los programadores respecto de determinados roles y por lo tanto a la pérdida de beneficios derivada de ello. Sin embargo -y lo que resulta más importante para nosotros-, como afirman acertadamente García y Steinmueller (2003), "Que la exclusión de tales roles debiera producir un movimiento de 'resistancia' no es sorprendente. Que este movimiento de resistencia parezca haber creado un punto de bifurcación en la organización y división del trabajo en la producción del software es mucho más notable". Es decir, que la resistencia a la apropiación exclusiva del software por parte de quienes detentan sus licencias haya derivado en una acción colectiva que articula una nueva forma de organizar el trabajo y la producción de software, y no sólo eso, ya que como dijimos antes cuestiona la propiedad privada sobre los programas mismos, no se explica en su totalidad por la reacción misma (e individual) de los informáticos, lo que hace necesaria la incorporación de otros determinantes -entre ellos, y a nuestro parecer, las relaciones de clase y de producción-, a la hora de explicar cabalmente el movimiento del software libre.

Pero, ¿qué significa software libre?. Para la *Free Software Foudation* (1999), creada entre otros por Richard Stallman en 1985, "'Software Libre' se refiere a la libertad de los

usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito (libertad 0); la libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1), el acceso al código fuente es una condición previa para esto; la libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2); la libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3), el acceso al código fuente es un requisito previo para esto". Una aclaración fundamental para entender esto, es que la palabra inglesa *free* no hace referencia a *gratuidad* sino a *libertad*, es decir al uso libre del *código fuente* del programa. Las empresas que producen software venden los programas o aplicaciones en forma *ejecutable*, es decir venden el código en forma *binaria* (en el nivel más bajo de programación, una secuencia indescifrable de ceros y unos), y lo que otorgan es el permiso o licencia para un uso preestablecido del programa, por lo tanto, el software queda protegido por un *copyright* que cristaliza una serie de derechos del autor (que en realidad es la empresa que lo fabrica) sobre el uso que los usuarios dan al mismo. Es más, en muchos casos, las empresas no sólo aseguran su propiedad por un producto concreto desarrollado por ellas (por ejemplo, el *Word* pertenece a *Microsoft*), sino que también y cada vez más se apropian de los *algoritmos*, es decir de la lógica de programación misma, que excede a un producto determinado (siguiendo el ejemplo anterior, *Microsoft* posee los derechos sobre los motores de ortografía y sinónimos que utiliza el *Word*, sobre la forma en que se realizan estos procesos). Entonces, el software libre busca lograr básicamente la libre disponibilidad del *código fuente* (el programa en su forma "lenguaje" de programación) lo que permitiría su estudio, modificación y posible mejora o adaptación a

las necesidades que surgen de cualquier usuario -que tenga la capacidad para hacerlo-, y además, la libertad de copiar y distribuir el código.

Ahora bien, antes de avanzar en cuestiones más concretas sobre el tema, queremos dejar señalados dos puntos que se refieren a ciertas consecuencias, que forman parte de la esfera *superestructural* que ha emergido sobre el desarrollo del software libre.

Por una parte, este movimiento práctico estuvo desde sus orígenes acompañado por un fuerte intento de construir una filosofía y una ética que lo sustenten no sólo subjetivamente, sino que a la vez sirvan de marco de legitimidad. Más allá de un posible análisis discursivo o ideológico de las mismas, es necesario tenerlas en cuenta a la hora de realizar un estudio exhaustivo del fenómeno y su desarrollo, ya que como afirma Marx (1974), son parte de las formas ideológicas en que los hombres adquieren conciencia del conflicto y luchan por resolverlo. Así, podemos ver que Stallman (1999b), al argumentar su oposición al software propietario afirma: "Los propietarios usan palabras difamatorias como 'piratería' y 'robo', al igual que terminología experta como 'propiedad intelectual' y 'daño', para sugerir una cierta línea de pensamiento al público, una analogía simplona entre los programas y los objetos físicos. Nuestras ideas e intuiciones acerca de la propiedad sobre los objetos materiales tratan acerca de si es justo *quitarle un objeto* a alguien. No se aplican directamente a *hacer una copia* de algo. Pero los propietarios nos piden que las apliquemos de todas formas". Si bien esto resulta un argumento antipropiedad selectivo, ya que refiere a un tipo específico de objeto -inmaterial y que se puede copiar-, no por ello habría que dejar de reconocer en él un problema que merece un tratamiento teórico, ya que en última instancia nos está remitiendo a las consecuencias que se derivan de la producción y

consumo de un tipo específico de producto, que cobra creciente relevancia en la fase capitalista actual, y además, a cómo (al menos un sector de) los sujetos productores del mismo buscan dirimir el conflicto que se articula en torno a la forma en *cómo* debe ser producido y consumido, es decir en el marco de las relaciones de propiedad.

Por otra parte, y en relación a lo anterior, podemos ver cómo se ha procurado que esta modalidad de producción de software logre afianzarse legalmente, y no sólo eso, que esa misma legalidad progresivamente "libere" todos los productos informáticos. Para ello la *Free Software Foundation* acuño el concepto de *copyleft*, en clara oposición a la idea de *copyright*: De acuerdo a Stallman (1999a), "El copyleft usa la ley de copyright, pero la da vuelta para servir a lo opuesto de su propósito usual: en lugar de ser un medio de privatizar el software, se transforma en un medio de mantener libre al software. La idea central del copyleft es que le damos a cualquiera el permiso para correr el programa, copiar el programa, modificar el programa y redistribuir versiones modificadas, pero no le damos permiso para agregar restricciones propias. De esta manera, las libertades cruciales que definen al 'software libre' quedan garantizadas para cualquiera que tenga una copia; se transforman en derechos inalienables". Utilizando la misma lógica de licencias y permisos, los pioneros del software libre han desarrollado la *General Public License* (GPL) en 1989. En palabras de Vidal (2000), "(l)a 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". Aunque el tipo de licencias ha ido proliferando y alguna de ellas no contienen la llamada cláusula "viral", la GPL es una de las más difundida. Esto, sin embargo, nos indica la heterogeneidad de intereses y concepciones que atraviesan al fenómeno de software libre -que se observa por ejemplo en la discusión en torno al uso del término *open source*-, y en el fondo, las relaciones sociales que se están tejiendo en torno a él. Sobre todo anotamos estas cuestiones pues habría que tenerlas en cuenta a la hora de analizar ciertas corrientes que depositan una enorme esperanza en el software libre por sus supuestos rasgos antagónicos con el capitalismo, ya que muchos ven en el *copyleft* , una nueva noción de bien público, no tutelado por el mercado ni por el estado, como un nuevo *espacio público no estatal*, incluso una nueva forma de comunismo o para algunos de anarquismo, cuestiones que a nuestro entender llevan a problemas del tipo de *qué* es un bien público, de si realmente que el software sea "libre" asegura un acceso público al mismo (ya que el soporte material -la computadora-, sigue siendo una mercancía y por lo tanto se necesita dinero para obtenerla), y si es posible bajo el capitalismo, en palabras de Engels, una "revolución silenciosa" de este estilo.

Pasemos ahora a tratar de entender por qué y cómo se ha tratado de interpretar este fenómeno, notando que en los análisis que presentaremos mayoritariamente se refieren al mismo como *open source*.

Este es un término que fue impulsado por uno de los "líderes" del movimiento del software libre, Eric Raymond, que lo incorporó en una de las revisiones a su clásico trabajo *La Catedral y el Bazar* -aparecido en 1998-, obra de referencia ineludible por parte de los que

tratan la temática. En ella, contrapone dos modelos de producción de software. Por un lado, está el modelo *catedral*, asociado a las corporaciones que producen software propietario (e incluso a la *FSF*), y que consiste en un estilo de desarrollo vinculado estrechamente a la ingeniería de software, fuertemente coordinado, centralizado y que sigue un riguroso plan. Por otro, el modelo *bazar*, cuyo ejemplo paradigmático es el sistema operativo *Linux*, que fue desarrollado voluntariamente (es decir sin retribución monetaria alguna) por miles de programadores de todo el mundo. Para Vidal (2000), Raymond resume al modelo bazar en tres máximas: 1) liberar rápido y a menudo; 2) distribuir responsabilidades y tareas todo lo posible; 3) ser abierto hasta la promiscuidad para estimular al máximo la cooperación. La metáfora del bazar tiene que ver con lo que sería un estilo de desarrollo sin mando, donde cada uno trabaja a su propia manera y siguiendo sus propios tiempos. Esto quizás podría haber sido una mera extravagancia de un grupo nutrido de *hackers* que detestaban a las grandes empresas de software, pero como señaló el mismo Raymond, "Linux es subversivo", ya que progesivamente reemplazó -y lo sigue haciendo- en proporciones considerables a lo que era el sistema operativo standard de las computadoras personales, el *Windows*, e incluso otro producto hecho con las mismas característica, el *Apache Web-server*, ha llegado a ser la principal aplicación utilizada para el manejo de redes informáticas.

Básicamente por estas razones, el estudio del *Linux* y de forma más general del open source, comenzó a ser un terreno fértil para el análisis y la aplicación de diversos modelos teóricos para explicar las características y alcances de este nuevo fenómeno.

Weber (2000) afirma que esencialmente son tres las cuestiones que provocan este interés en los cientistas sociales: 1) las motivaciones de los individuos, es decir, cómo explicar que programadores altamente calificados y talentosos elijan dar una parte sustancial de su tiempo e ideas para un proyecto sin recibir compensación; 2) la coordinación, o cómo se articulan los sujetos sin mediar una autoridad empresarial ni un mecanismo de precios; 3) la complejidad, en relación a de qué manera se logra manejar la división del trabajo en una comunidad dispersa y en constante incremento, es decir creciente en complejidad y produciendo un bien complejo como es un software.

El tema de la *motivación* o incentivo que acerca a los programadores al diseño de un "bien público", es un tema recurrente en los estudios sobre el software libre y ha sido tratado por varios autores, en los cuales podemos encontrar tanto similitudes como diferencias surgidas a partir de los progresivos análisis realizados, pero en general la mayoría se mueven en un sustrato común (aunque no siempre expreso), que es el par conceptual *costo/beneficio*, que posee la fuerte impronta de una concepción de los sujetos como *homo economicus*. Es de notar que dicho supuesto actúa más allá de que efectivamente se tenga una remuneración, ya que como lo demuestra el estudio de Kim (2003) los porcentajes de los que adhieren a un mismo tipo de motivación, son prácticamente los mismos sean los entrevistados pagos o no pagos. Raymond (1998) encuentra tres motivos básicos para la participación en este tipo de proyectos: a) porque los programadores pueden beneficiarse directamente usándolo para ellos mismos; b) porque disfrutan (*enjoy*) del trabajo de programación mismo; c) porque ellos pueden ganar reputación ante sus pares haciendo contribuciones de alta calidad. Herrman, Hertel y Niedner (2000) y Lakhani y Wolf (2001), coinciden con esta caracterización, aunque encuentran que mejorar la reputación se verifica como un motivo

significativamente bajo en los estudios que realizaron, si bien reconocen que puede ser debido a cuestiones de prejuicio por parte de los entrevistados. Lerner y Tirole (2000), al contrario, refuerzan la importancia de la reputación, a la que consideran un *ego gratification incentive*, y que junto al *career concern incentive* -que refiere a ofertas futuras de trabajo en empresas o *joint ventures*-, serían los principales motivos de la participación en proyectos open source, aunque aclaran que probablemente la mayoría de los programadores responden a ambos incentivos, superando las visiones que toman a los mismos de forma excluyente. Kollock (1999) se inclina por una visión que relaciona los incentivos con beneficios que no son directos. Si bien la reputación también queda incluida en ellos, el autor agrega: a) un sentido de "eficacia" en los participantes; b) la búsqueda del bien del grupo de participantes mismo; c) la expectativa de reciprocidad. Aunque aquí podemos ver un agregado tanto de motivos *individualistas* como *altruistas* -clasificación que se transformará en otro punto de discusión en la literatura consultada-, la cuestión de la reciprocidad dará lugar a toda una línea interpretativa del fenómeno ligada a la llamada *gift economy* o economía de regalos o dones (de bienes de alta tecnología), que derivará en una discusión en torno a las características de la misma, ya que si bien el mismo Kollock (1999) la plantea como una economía opuesta a las relaciones de mercado, Barbrook (1998), sin dejar de reconocer esto, piensa que dicha forma sólo puede expandirse por una colaboración mutua con el sector comercial y por lo tanto en realidad habría una "economía digital mixta", y Ljungberg (2000) afirma que si bien parece oponerse a las relaciones mercantiles, muchos de sus aspectos no son tan diferentes.

El problema de la *coordinación* es otro campo fértil para el análisis social y está en parte ligado al tema de las motivaciones, ya que muchos de los modelos aplicados las suponen

como básicas. Este es el caso de la aplicación de la *teoría de los juegos*, cuyo objeto es el análisis de las interrelaciones entre dos o más individuos (o grupos) y la búsqueda de un modelo de actuación óptimo. En el caso del software libre encontramos trabajos como los de Martínez (1999) y Vidal (2000), donde a través de la utilización de los modelos del *dilema del prisionero* o del *Tit for Tat* (pagar con la misma moneda), se puede verificar la posibilidad y existencia de cooperación productiva, más allá de que las motivaciones sean fundamentalmente individuales (cooperación egoísta) o se articulen con motivos altruistas. Otros autores, también teniendo como base el tema motivacional, han buscado ensayar explicaciones a partir de la utilización de teorías de la *acción colectiva*, o de lo que son uno de sus casos particulares, los *movimientos sociales*. Hertel, Nieder y Herrman (2003) definen como movimiento social a los "esfuerzos de un gran número de personas por resolver colectivamente un problema que tienen en común", y aclaran que si bien las comunidades open source no son un típico movimiento social, algunos de sus objetivos políticos y sociales pueden ser entendidos como un esfuerzo colectivo por resolver problemas comunes, y que incluso, los motivos subyacentes de aquellos que contribuyen son similares a los de los que participan en movimientos sociales. Esta línea de análisis ha complejizado la interpretación del fenómeno, ya que por ejemplo, en el caso de estos autores se reformula la batería de motivaciones, "nos referimos a cuatro componentes motivacionales para explicar la acción voluntaria en los movimientos sociales: motivos colectivos, motivos orientados por normas, motivos por recompensas y procesos de identificación".

El análisis de la coordinación en el software libre ha llegado a cuestionar profundamente interpretaciones tempranas sobre el mismo, de manera fundamental al "modelo bazar"

propuesto por Raymond (1998). La idea de la cooperación sin jerarquías ha sido fuertemente críticada. Healy y Schusmann (2003) enfatizan dos cuestiones a partir de su análisis empírico del repositorio de proyectos *Sourceforge*: una, que el rol de los líderes de proyectos para desarrollar la movilización de los programadores es fundamental dada la participación voluntaria, y la otra, que la organización jerárquica es de vital importancia, verificando que los proyectos exitosos tendieron a poseer un fuerte componente jerárquico. Ljungberg (2000) señala que uno de los modelos más comunes de coordinación de proyectos open source es el desarrollado bajo un *benevolent dictator*, en general un gran y talentoso programador, que actúa como un juez, teniendo derecho a tomar las decisiones centrales y acreditar a los participantes imparcialmente. Aunque también existen alternativas, la *rotation dictatorship* (utilizada para el desarrollo del lenguaje de programación *Perl*) o el *voting committe* con co-desarrolladores (usado por ejemplo en el *Apache Web-server*). García y Steinmuller (2003) se inclinan por un modelo de análisis de *distributed authority* como principal medio aplicado por los programadores no sólo para dirigir el proyecto sino también para incrementar su estabilidad, evaluar la calidad de las contribuciones y disminuir la severidad de los conflictos que se producen. Su análisis tiene la virtud de reconocer frente a las interpretaciones más frecuentes, que de forma abstracta e idealista plantean el fenómeno del open source desde una perspectiva de "intereses comunes", la diversidad de los mismos.

IV

Como se ha podido observar hasta aquí, los análisis del fenómeno del software libre se orientan en general por modelos fundamentalmente *ahistóricos* y que sacrifican las cuestiones estructurales por interacciones y motivos individuales, sean estos egoístas o altruistas. La única crítica hallada en la bibliografía ha sido la de Weber (2000), quien acertadamente afirma que "la macro-lógica del *open source* no se reduce a una agregación de las motivaciones de los individuos que participan". Creemos que esta es una buena idea a tener en cuenta para penetrar en la complejidad del fenómeno, no sólo teóricamente sino también metodológicamente. Si bien las motivaciones individuales son un elemento importante para explorar la subjetividad, están por un lado, como categoría, impregnadas por una fuerte idea instrumental de la acción, y por otro, no debemos olvidar que los motivos son en gran medida sólo una enunciación del sujeto que tiende a la justificación de sus acciones.

En este sentido, pensamos que uno de los puntos habría que tener en cuenta para explorar y profundizar es la forma de articulación o forma en que se ponen en juego dos polos de una relación, lo que podríamos llamar la dimensión *conciente* del fenómeno, con lo que se estructura a espaldas de los sujetos, en cierta forma lo que Marx (1988) indica en su conocida frase "(n)o lo saben, pero lo *hacen*". Esto resulta de vital importancia ya que el software libre es caracterizado en general como una forma alternativa, ya sea de compartir conocimiento, de producir software, de producir en base a comunidades, de organización del trabajo basado en el conocimiento, o como también de nuevo modelo de negocios o de relaciones con el cliente, como lo expresan Ljungberg (2000) y Weber (2000). Esto nos remite al aspecto material del fenómeno, ya que en el plano económico el software libre apunta en principio a que los programadores vivan no de producir un bien informático para

ganar con la venta de una mercancía privada y protegida con licencias, sino que se ofrezca mantenimiento y soporte técnico de un producto básicamente "libre". Sin embargo, la evolución del fenómeno lo ha ido desplazándo como altenativa radical, ya que como afirma Söderberg (2002), "el antagonismo entre el sofware libre y el software propietario puede ser cuestionado en cuento recordamos que las grandes corporaciones están ahora respaldando al Open Source", reconociendo las inversiones y estrategias desarrolladas en los últimos años por parte de empresas como *IBM*, *Sun Microsystems*, *Oracle*, etc.

Por ello y retomando lo dicho más arriba, pensamos que una adecuada línea de análisis sería interpretar el fenómeno incluyendo de forma privilegiada las dimensiones de las *relaciones de producción* y de las *relaciones de clase*, que puestas en un marco histórico y dinámico, y a manera de hipótesis, permitirán interpretar el devenir del software libre u open source (nombre que surge precisamente como estrategia de *marketing*) en relación al conflicto entre trabajo y capital. Sin embargo, el mayor desafío sería evitar una *fetichización*, y por lo tanto una *unilateralización* o *reduccionismo* de uno de los polos planteados en el párrafo anterior, es decir del polo *subjetivo* que nos haría caer en una suerte de voluntarismo abstracto, o del polo *objetivo* que llevaría a una forma de *estructuralismo sin sujeto*.

Nota: Las citas de textos en inglés son traducciones propias. Se conservan algunos términos del original en cursiva.

Bibliografía citada

- Antunes, R. (1999): *¿Adiós al trabajo? Ensayo sobre las metamorfosis y el rol central del mundo del trabajo*, Buenos Aires, Editorial Antídoto, Colección Herramienta.
- Barbrook, R. (1998): "The Hi-Tech Gift Economy", *First Monda*y, en: http://www.firstmonday.org/issues/issue3_12/barbrook/index.html, versión en castellano en: http://www.rcci.net/globalizacion/llega/fg108.htm
- Barrett, R. (2001): "Labouring under an illusion? The labour process of software development in the Australian Information Industry", *New Technology, Work and Employment,* Vol. 16, N° 1.
- Beirne, M., Ramsay, H., y Panteli, A. (1998): "Developments in computing work: Control and contradiction in the software labour process", en Thompson, P. y Warhurst, C. (Eds.): *Workplaces of the Future.* Houndsmill, UK, Macmillan Business.
- Braverman, H. (1987): *Trabajo y capital monopolista*. México, Editorial Nuestro Tiempo.
- de la Garza Toledo, E. (2001): "Problemas clásicos y actuales de la crisis del trabajo", en de la Garza Toledo, E. y Neffa, J. (Comps.): *El futuro del trabajo. El trabajo del futuro*, Buenos Aires, CLACSO.
- Free Software Foundation (1999): "La definición de Software Libre", en: http://www.gnu.org/philosophy/free-sw.es.html

- García, J. M. y Steinmueller W. (2003): "The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?", Science and Technology Policy Research, University of Sussex, INK Research Working Paper N° 1.
- Gorz, A. (1982): *Adiós al proletariado (Más allá del socialismo)*. Barcelona, El viejo topo.
- Greenbaum, J. (1995): *Windows on the Workplace*. New York, Montlhy Review Press.
- Hannemyr, G. (1999): "Technology and Pleasure. Considering Hacking Construstrive Self-Selection Strategies for Information Goods", *First Monday*, Vol. 4 N,º 2, en: http://firstmonday.org/issues/issue4_2/hannemyr/index.html
- Healy K. y Schussman, A. (2003): "The Ecology of Open-Source Software Development", Working Paper, Open Source Community, MIT, en: http://opensource.mit.edu/papers/healyschussman.pdf
- Herrmann, S., Hertel, G. and Niedner, S. (2000): "Linux Study Homepage", http://www.psychologie.uni-kiel.de/linux-study/
- Hertel, G., Niedner, S. y Herrmann, S. (2003): "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel", Research Policy Special Issue Working Paper, en: http://opensource.mit.edu/papers/rp-hertelniednerherrmann.pdf
- Katz, C. (1996): "El carácter de las nuevas tecnologías de la información", *Acheronta*, Revista de Psicoanálisis y Cultura, N° 3, en: http://www.acheronta.org
- Kim, E. (2003): "An Introduction to Open Source Communities", Blue Oxen Associates BOA-00007 en: http://opensource.mit.edu/papers/blueoxen.pdf
- Kogut, B. y Metiu A. (2001): "Open-Source Software Development and Distributed Innovation", *Oxford Review of Economic Policy,* Vol. 17, Nº 2.

- Kollock, P. (1999): "The Economics of Online Cooperation: gifts and public goods in cyberspace", en Smith, M. and Kollock, P. (Eds.): *Communities in Cyberspace*, Routledge, London.
- Kraft, P. y Dubnoff, S. (1986): "Job content, fragmentation and control in software work", *Industrial Relation*s, Vol. 25, Nº 2.
- Lakhani, K. y Wolf, R. (2001): "Does Free Software Mean Free Labor? Characteristics of Participants in Open Source Communities", BCG Survey Report, Boston Consulting Group Report, Boston, MA, en: http://www.osdn.com/bcg/
- Lerner, J. y Tirole, J. (2000): "The Simple Economics of Open Source", National Bureau of Economic Research (NBER) Working Paper 7600, en: http://www.nber.org/papers/w7600
- Levinson, M. (2001): "IT Workers of the World: Are They Uniting", *CIO Magazine*, en: http://www.newecon.org/www.cio.com/archive/050101/world_content.html
- Ljungberg, J. (2000): "Open Source Movements as a Model for Organizing", *European Journal of Information Systems*, Vol. 9, Nº 4, en http://www.viktoria.informatik.gu.se/groups/KnowledgeManagement/Documents/ecis2000.pdf
- Lockyer, C., Marks, A y Mulvey, G. (2001a): "Programming teamwork? Evidence from the Scottish software industry", *5th International Workshop on Teamworking*, Catholic University of Leuven, en http://www.strath.ac.uk/Other/futureofwork/publications_pdf files/IWOT 2001 programming teamwork.PDF
- Lockyer, C., Marks, A. y Mulvey, G. (2001b): "In need of a new language? Issues of identity in software development teams", *19th International Labour Process Conference*, University of London, en

http://www.strath.ac.uk/Other/futureofwork/publications pdf files/LP 2001 software identity.PDF

- Martínez, J. (1999): Software Libre: una aproximación desde la Teoría de Juegos, en: http://www.dit.upm.es/~jantonio/documentos/revistas/teoriajuegos/ teoriajuegos.html
- Marx, K. (1988): *El capital*. México, Siglo XXI. 3 Tomos, 8 Volúmenes.
- Marx, K. y Engels, F. (1974): *Obras Escogidas*. Moscú, Editorial Progreso. 3 Tomos.
- Orlikowski, W. (1988): "The Data Processing Occupation: Professionalisation or Proletarianisation?", *Research in the Sociology of Work*, N° 4.
- Raymond, E. (1998): "La Catedral y el Bazar", en: http://www.sindominio.net/biblioweb/ telematica/catedral.pdf
- Rifkin J. (1996): *El fin del trabajo. Nuevas tecnologías contra puestos de trabajo: el nacimiento de una nueva era*. Barcelona, Paidós.
- Söderberg, J. (2002): "Copyleft Vs. Copyright: A Marxist Critique", *First Monday*, Vol. 7, Nº 3, en: http://www.firstmonday.dk/issues/issue7_3/soderberg/
- Stallman, R. (1999a): "El Proyecto GNU", en: http://www.gnu.org/gnu/ thegnuproject.es.html
- Stallman, R. (1999b): "Por qué el software no debe tener propietarios", en: http:// www.gnu.org/philosophy/why-free.es.html
- Union Network International (2001): "La sindicalización en la economía mundial cambiante - 1er Congreso de la UNI", en: http://www.union-network.org/UNIsite/Sectors/IBITS/IBITS.html
- Vidal, M. (2000): "Cooperación sin mando: una introducción al software libre", en: http://www.sindominio.net/biblioweb/telematica/softlibre/

- Weber, S. (2000): "The Political Economy of Open Source Software", BRIE Working Paper 140, E-conomy Project Working Paper 15, en: http://brie.berkeley.edu/~briewww/ pubs/wp/wp140.pdf