FACULTAD LATINOAMERICANA DE CIENCIAS SOCIALES
SEDE ACADÉMICA MÉXICO

MAESTRÍA EN CIENCIAS SOCIALES
XVII PROMOCIÓN
2008-2010

Nuevas formas de valorización del conocimiento en el esquema de realización de ganancia del capital.

Caso: Asociación Mexicana Empresarial del Software Libre

Tesis que para obtener el grado de Maestro en Ciencias Sociales
Presenta:

Alfonso Cano Robles

Director de tesis:
Dra. Úrsula Zurita Rivera

Seminario de tesis:
Economía y Sociedad del Conocimiento

México, D. F. Agosto de 2010

Este posgrado fue cursado gracias a una beca otorgada por
el Consejo Nacional de Ciencia y Tecnología (CONACYT) del Gobierno de México

FLACSO
MÉXICO

FACULTAD LATINOAMERICANA DE CIENCIAS SOCIALES SEDE ACADÉMICA MÉXICO

MAESTRÍA EN CIENCIAS SOCIALES

XVII PROMOCIÓN

2008 – 2010

Título de tesis

Nuevas formas de valorización del conocimiento en el esquema de realización de ganancia del capital.

Caso: Asociación Mexicana Empresarial del Software Libre

Tesis que para obtener el grado de Maestro en Ciencias Sociales

Presenta:

Alfonso Cano Robles

Director de tesis:

Dra. Úrsula Zurita Rivera

Seminario de tesis

Economía y Sociedad del Conocimiento

México, D. F. Agosto de 2010

Índice de contenido

Índice de tablas

Índice de ilustraciones

Resumen

Para hacer evidente las dinámicas que rigen la sociedad capitalista del conocimiento. Se observa una realidad en movimiento que presenta nuevas formas de valorización y de realización de ganancia. A partir de estas categorías de análisis, se observa la realidad concreta que presenta la industria del software en general y del contexto mexicano en particular ante la práctica social del software libre en la Asociación Mexicana Empresarial del software libre (AMESOL). Para ello, se hace uso de la concepción del software en general, como un conocimiento objetivado. Y del software libre en particular como un "bien club", es decir, como conocimiento codificado.

Para lograr lo anterior, y desde un enfoque crítico cuyo sustento epistemológico es la reconstrucción articulada de la realidad, se parte de una contextualización en la que se presta especial atención a la valorización y la realización de la ganancia. Y que se encuentra regido por la relación que guarda trabajo-conocimiento, el uso característico de la tecnología y la restricción del conocimiento. Donde se considera que la lógica del capitalismo delinea un doble movimiento que se reconfigura continuamente, gobernado por intereses particulares y las constantes innovaciones técnicas que también destruyen lo viejo.

Abstract

To make evident the inherent dynamics of the capitalist society of knowledge. This work beholds a moving reality which presents new ways of granting value (as worth) and materialize profit. From these categories of analysis, a concrete reality is watched at the software industry in general and the Mexican context in particular towards the social practice of free (libre) software at the Asociación Mexicana Empresarial del software libre (AMESOL). To do that, it is precise to use the conception of software as an objective knowledge. And that of free (libre) software as a “good club”, in other words, as codified knowledge.

To accomplish the precedent, and form a critic scope which epistemic support is an articulated reconstruction of reality, first is made a contextualization that presents special attention to the way value is granted (as worth) and the materialization of profit. Which is governed by the relationship between work-knowledge, the characteristic use of technology and the restriction of knowledge. Where is considered that the logic of capitalism draws a double movement that reconfigures continuously, given particular interests and the constant technical innovations that also destroys old things.

A Teresa, mis padres y hermano
con amor

A Rogelio por ser un guía de vida

AGRADECIMIENTOS

A la Dra. Úrsula Zurita Rivera
y su valiosa asesoría a lo largo del proceso.

A mis lectores:
la Dra. Mónica Casalet Revenna,
el Dr. Prudencio Óscar Mochi Alemán
y al Mtro. Edgar Buenrostro Mercado
por su atención y ayuda.

Al Dr. Juan Cristóbal Cobo Romaní
y al Mtro. Leonel González González
por su guía.

Al Consejo Nacional de Ciencia y Tecnología
(CONACYT)
por hacer posible cursar este postgrado
y cuyo fruto es este trabajo.

Al Ing. Daniel Ceballos Lugo
por brindarme su tiempo y amistad.

A los 33 compañeros del postgrado.

A todos los que de alguna forma,
lo hicieron posible.

Introducción

Esta indagación busca hacer evidentes los rasgos que componen la realidad que han hecho posibles nuevas formas de valorización del conocimiento y analizar la relación que guardan éstas con las formas de realizar ganancia dentro de una sociedad capitalista. Estos rasgos se han encontrado íntimamente ligados a la aparición de las tecnologías de la información y la comunicación en los procesos productivos. Lo anterior ha permitido que el conocimiento se consolide como el factor más importante de la producción pero también, que adquiera otras propiedades en tanto se ha objetivado como conocimiento codificado al que se ha llamado *software* (véase: Capítulo I).

En la sociedad del conocimiento, han emergido nuevas tecnologías que permiten comunicar información y procesarla en tiempo real. Para ello se han desarrollado computadoras[1] *(hardware)* y junto con ellas textos que les permiten funcionar, a esto se le ha designado software. El software que hoy permite, como cualquier otro texto ser estudiado, modificado y redistribuido, se le llama software libre. Al hacerlo, no se viola ningún derecho de autor, pues las licencias bajo las que se hacen accesibles estos programas, brindan la autorización expresa que faculta a todo aquel que lo obtiene para realizar dichas acciones.

Ahora bien, existen dos aspectos que hacen particularmente interesante a este tipo de software. La primera es que a diferencia de otros, permite aprender de él y convertirlo en parte del acervo de conocimientos a mano, de cualquier persona que lo pueda leer y comprender. La segunda, es que es posible hacer negocio con él, es decir, realizar ganancias. De donde salta la pregunta ¿cómo es posible realizar ganancias a partir de aquello que es de libre acceso?

[1] Por computadoras, no se hace referencia sólo a la computadora personal de escritorio o portátil, sino a todo dispositivo electrónico que requiere *software* para funcionar integrado a partir de microprocesadores. Como pueden ser: equipos de reproductores de música portátiles, el control y temporizador de lavadoras, teléfonos celulares, diversos dispositivos en el automóvil (frenos con sistema antibloqueo, supervisión de fallas, asistencia de cambios de velocidad, etc.).

Hoy existen miles de proyectos de software libre, y continuamente aparecen nuevos (véase: Capítulo II). A lo largo de este trabajo se esbozaran las condiciones históricas que han hecho posible la emergencia de este tipo de software, dichas condiciones proporcionan el marco analítico que permite explicar este fenómeno a partir de una reconstrucción articulada de la realidad en un caso concreto.

1. El contexto problemático

Para enmarcar el planteamiento del problema de esta investigación este apartado se ha subdividido en dos partes. En la primera se aborda la relevancia del software en una realidad concreta, como es el ámbito mexicano. Y posteriormente, se hace una pequeña discusión de los ejes centrales que han guiado los temas de discusión sobre el software libre. En este último punto, se hace el planteamiento del ámbito problemático en este contexto teórico, en torno a la necesidad de analizar el software libre como una forma de conocimiento a partir de la cual se realizan ganancias.

a. El software en México y su contexto internacional

Para hablar del software libre, es necesario hablar de cómo es que se clasifica y se mide la industria del software. Ya que, además de brindar un panorama general, ofrece la posibilidad de resaltar el potencial que tiene el software libre para la realidad mexicana. Por ello, en este apartado se toma como referente la evaluación que nos ofrece principalmente la World Information Technology and Services Alliance (WITSA, 2006), para ubicar a México en el panorama internacional, prestando atención a su desempeño en el ámbito latinoamericano. Luego, pasaremos discutir las conclusiones que se derivan de la lectura de estos datos y el significado que tienen para la industria en México ante el potencial que ofrece el software libre.

Para esto, se debe tener en cuenta que la industria del software es cuantificada como parte de las llamadas Tecnologías de la Información y la Comunicación (TIC) que forman parte del Sector Electrónico Informático (SE-I[2]). Este sector, se haya integrado por rubros como son las comunicaciones, software, hardware y servicios de computación.

El conjunto del SE-I, se encuentra altamente diferenciado en México, ya que a la parte de comunicaciones le corresponde el 66% del gasto al concretarse en 17 000 millones de dólares, donde el software apenas alcanza el 3%, con 776.5 millones de dólares. También, como parte de este sector, se encuentran segmentos que se pueden catalogar como medianos, tal es el caso del rubro del hardware, que alcanza 5 076.3 millones de dólares, 20% del mercado y los servicios de computación que alcanzan 2 709.3 y se coloca con el 10.68% de gastos (Mochi & Hualde, 2006).

Así, el mercado del software a nivel internacional, contabilizado dentro de la industria de las TIC, alcanzó 288 806 millones de dólares, en el cual tuvo un crecimiento de 69% en el quinquenio que comprende al periodo 2001-2005. Donde el mercado es dominado por Estados Unidos, país al que corresponde el 44% del total mundial. Para poner esta cantidad en perspectiva, se puede aseverar que el conjunto de países que conforman Europa Occidental[3] alcanzaron para el 2005, una cifra un poco menor a 100 000 millones de dólares, inferior a los 125 847.2 millones de dólares que se ganaron en Estados Unidos (WITSA, 2006) (Tabla 1).

En contraste, México tenía el lugar 22 en el mercado de software, con 548.6 millones de dólares que corresponde al 0.29% de la producción internacional en el año 2001. Para el año 2005, México tenía un mercado de software con un valor ya mencionado de 776.5 millones de dólares que correspondió al .26% de la producción de ese año, lo que significó que en el

[2] Siguiendo a Ordóñez (2006), el término SE-I parece ser el adecuado para designar al sector de la industria conocido como “Tecnologías de la información y las Comunicaciones” (TIC), tal como lo denomina la OECD o el Foro Económico Mundial (WEF, por sus siglas en inglés). Ya que también, permite incluir a los productos y servicios que provee, así como a la industria dedicada al circuito integrado y la digitalización.

[3] Los países considerados son: Alemania, Reino Unido, Francia, Italia, P. Bajos, España, Suiza, Suecia, Bélgica, Dinamarca, Austria, Noruega y Finlandia.

periodo 2001-2005, creció 54%, con proyecciones de crecimiento de hasta 845.1 millones de dólares para el 2006 (WITSA, 2006) (Tabla 1).

Pese a ello, el crecimiento de México fue inferior al de otros países de América Latina, que crecieron en promedio 106% e incluso, bajó su posición en el escalafón internacional al colocarse en el lugar número 27 en ese año. En contraste, Brasil vio un incremento en su producción en 125%, e incluso en mercados más pequeños, como es el caso colombiano, se creció 119% (Mochi & Hualde, 2006). Donde en cifras absolutas, México tiene el segundo mercado más grande de software, atrás de Brasil (Tabla 2 e Ilustración 1).

TABLA 1: El mercado mundial de software, 2001-2006[a]
(En millones de dólares)

#	País	2001	2002	2003	2004	2005	2006	% de crecimiento
1	**Estados Unidos**	98103.3	97204.4	104918.3	115568.1	125847.2	135655.1	38.3
2	**Alemania**	13127	14527.4	18380	21317.7	22358.3	23802.3	81.3
3	**Reino Unido**	12315.8	13472	16466.9	20225.9	21576.3	22650.3	83.9
4	**Japón**	13435.8	13100.3	14719.9	16810.4	17014.1	17812.8	32.6
5	**Francia**	9205.5	10096.6	12908.5	15365.9	16432.4	17675.4	92
6	**Italia**	4747.7	5146.8	6598.1	7828	8363.7	8964.7	88.8
7	**China**	1657.9	2252.6	3344.4	5295.4	7939.5	11376	586.2
8	**Canadá**	4185.2	4297.2	5259.6	6263.2	7213.2	8007.6	91.3
9	**P. Bajos**	3805.6	4157.6	5239.5	6084.5	6495.3	7191.4	89
10	**Australia**	2398.3	2698.5	3621.8	4594.7	5556.2	6605.9	175.4
11	**España**	2009.1	2241.8	2919	3459.8	3877.1	4379.3	118
12	**Suiza**	2231.1	2467.6	2986.2	3495.9	3853.6	4432.4	98.7
13	**Brasil**	1697.5	1786.8	2468.7	2876.7	3565.9	3827.9	125.5
14	**Suecia**	1980.1	2179.6	2935.3	3352.6	3443.7	3757.6	89.8
15	**Bélgica**	1392.5	1548.7	2002.7	2414.6	2741.7	3132.5	125
16	**Sudáfrica**	723.8	800.4	1327.7	1965.1	2368.5	2780.5	284.2
17	**República de Corea**	918.6	1115.8	1355.9	1733.7	2261.5	2848.5	210.1
18	**Dinamarca**	1205.9	1344.7	1707.8	2012.5	2229.9	2517	108.7

(Continúa)

Tabla 1 *(Continuación)*: El mercado mundial de software, 2001-2006[a]
(En millones de dólares)

#	País	2001	2002	2003	2004	2005	2006	% de crecimiento
19	**Austria**	1170.7	1291.4	1676.8	1993	2200.2	2466.8	110.7
20	**India**	455.9	588.3	947.5	1350.4	1907.7	2518.6	452.4
21	**Noruega**	981.1	1100.6	1268.1	1549.8	1900.8	2203.9	124.6
22	**Finlandia**	932.7	1039.3	1317.9	1565.7	1625.9	1844.4	97.7
23	**Turquía**	217.3	283.8	492.8	698.4	939.3	1178	442.1
24	**Rusia**	394.7	450.1	570.2	742.2	923	1055.5	167.4
25	**República Checa**	334.3	380.4	531.1	701.8	859.5	1120.6	235.2
26	**Polonia**	471.8	537.3	653.3	733	828.6	968.3	105.2
27	**México**	548.6	610.8	639.7	692.1	776.5	845.1	54
28	**Israel**	527.2	522.3	569.3	653	700.9	767.6	45.6
29	**Argentina**	371.9	379.7	430.5	488.9	571.1	626.9	68.6
30	**Hong Kong**	317.5	358	373.4	431.6	492.1	558.3	75.8
31	**Colombia**	167.8	175.4	200.7	250.4	334.4	368.4	119.5
32	**Arabia Saudita**	185.4	196	209.4	261.5	306.1	322.1	73.7
33	**Venezuela**	162.4	176.8	184.3	220.7	249.1	313.5	93
34	**Vietnam**	55.8	71.7	99	138.1	215	300.5	438.5
35	**Chile**	100.6	101.9	117.8	145.4	172.5	190.5	89.4
36	**Panamá**	110.7	113.9	128.6	145.5	160.5	175.3	58.4
37	**Costarica**	74.3	80.3	89.2	96.2	105.2	112.7	51.7
38	**Nigeria**	17.5	25	31.9	47.5	57.4	71.9	310.9
39	**Uruguay**	36.2	42.7	34.8	39.1	51	57.9	59.9
	Resto del mundo	5017.1	5669.4	7006.7	8695.4	10291.6	12082.6	140.8
	Total	**187792.2**	**194633.9**	**226733.3**	**262304.4**	**288806.5**	**317566.6**	**69.1**

Fuente: Elaboración propia a partir de WITSA (2006).

a. Los datos del año 2006 son proyecciones, por lo que la tabla se encuentra ordenada de acuerdo al año 2005.

Tabla 2: El mercado de software en América Latina en el periodo 2001-2005[a]
(En millones de dólares)

País	2001	2002	2003	2004	2005
Argentina	371.9	379.7	430.5	488.9	571.1
Brasil	1697.5	1786.8	2468.7	2876.7	3565.9
Chile	100.6	101.9	117.8	145.4	172.5
Colombia	167.8	175.4	200.7	250.4	334.4
Costarica	74.3	80.3	89.2	96.2	105.2
México	548.6	610.8	639.7	692.1	776.5
Panamá	110.7	113.9	128.6	145.5	160.5
Uruguay	36.2	42.7	34.8	39.1	51
Venezuela	162.4	176.8	184.3	220.7	249.1
Otros	214.8	227.6	278.9	319.4	365.8
Totales	**3484.8**	**3695.9**	**4573.2**	**5274.4**	**6352**

Fuente: Elaboración propia a partir de datos de WITSA (2006).

a. En las estimaciones originales de WITSA (2006), se contabiliza a México dentro del bloque "América del Norte". Así como, el rubro de "Otros", fue calculado a partir de la diferencia de la adición de los países y los totales que presenta esta base de datos para el bloque "América Latina".

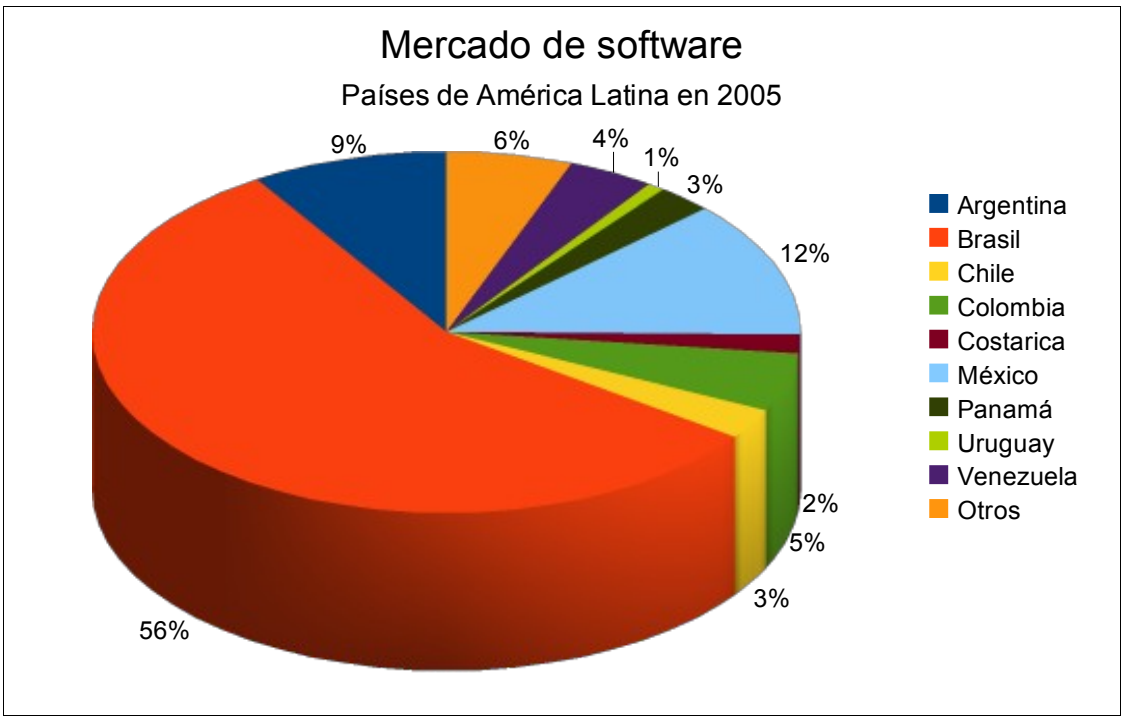

Ilustración 1: El mercado de software en América Latina en 2005

Fuente: Elaboración propia a partir de datos de WITSA (2006).

Ante esta cuantificación, se debe tener presente la forma en que se mide la industria del software, ya que es posible que haya subvaloración de la misma. Esto se debe a que el software puede ser producido para usos y formas de consumo que escapan a la fiscalización. Según la tipología que nos presenta Prudencio Mochi (2006), el núcleo de la industria del software se encuentra dividida en: a) software empaquetado[4], b) los servicios (que incluyen desarrollo de software a la medida, consultoría, integración de aplicaciones empresariales y otros servicios) y c) el software incrustado o embebido (*firmware*[5]). Así, el software que es producido para el consumo interno de una empresa o para ser incrustado en bienes de consumo como firmware, puede que escapen a las formas de supervisión fiscal.

De esta forma Prudencio Mochi y Alfredo Hualde (2009), nos dicen que la industria mexicana del software tiene cuatro categorías fundamentales:

1. *Industria nacional de software y servicios informáticos*: se caracteriza por estar constituida por pequeñas y medianas empresas desarrolladoras que se encuentran más orientadas a los servicios que al llamado software empaquetado. De esta forma, pueden subcontratarlos para producir software integrado a la industria electrónica o para la exportación directa en cadenas internacionales. En la categoría de los servicios, en el año 2005 el precio del mercado nacional fue de 2 311 millones de dólares y para el rubro de software empaquetado ascendía a 817 millones de dólares.
2. *Producción interna (consumo propio interno)*: integrada por la producción propia orientada al autoconsumo especializado (*in-house*), para el cual también habilitan servicios. Aquí es donde reside una oportunidad de desarrollo para la industria del software, si esta tarea comienza a ser

4 El software empaquetado es aquel que se encuentra orientado al consumo masivo a través de diversos distribuidores.

5 El firmware es un software, que se puede entender como un conjunto de instrucciones programadas en un dispositivo hardware. Y el cual contiene la instrucción necesaria para que dicho dispositivo se comunique con otro, o bien, para desempeñar más eficientemente ciertas operaciones, propias del fin para el que fue construido el equipo. El firmware, de un dispositivo se suele almacenar en memoria sólo para lectura (ROM, por sus siglas en inglés), o bien, en memoria tipo flash-ROM, ésta última permanece sin cambios a menos que sea actualizada.

practicada por subcontratistas. Este sector se estima a partir del costo de nómina del área de sistemas y del número de empleados que hacen desarrollo y planteamiento de software. El cómputo de lo anterior se estima en 1 738 millones de dólares, así se advierte el poco desarrollo que tiene el sector especializado y que puede ser desarrollado.

3. *Filiales de las grandes empresas transnacionales de software empaquetado*: Distribuyen este tipo de software en el país y realizan las labores de soporte técnico y asistencia para grandes empresas como Microsoft, SAP, IBM, Oracle, HP, etc.
4. *Grandes empresas transnacionales exportadoras de productos electrónicos*: Es la producción de software para ser embebido en productos electrónicos, los cuales son producidos para exportar, como es el caso de INTEL y HP. E incluso para exportar directamente el software, como es el caso de IBM. En México, este tipo de software se produce sobre todo en el conglomerado abierto o sin fronteras de Guadalajara, el cual facturó 130 millones de software integrado (firmware) lo que representa cerca del 84% del total de software embebido que se genera en México. Ya que en Guadalajara existe 22 empresas donde se diseña tanto el hardware como el software que lo opera y sólo existen otras 3 de estas empresas en México, las cuales se encuentran en la región de Baja California.

En el país, las empresas que dominan la distribución de software empaquetado son transnacionales como Microsoft, SAP, IBM y Oracle con excepción de Aspel que es mexicana. Sin embargo, en el rubro de desarrollo del software en México destacan empresas nacionales como Hildebrando y Softtek, al alcanzar en el 2005: 47.75 y 33.78 millones de dólares, para al colocarse en primero y segundo lugar respectivamente en cuanto a facturación a nivel nacional. Así, de las 32 empresas que presenta Select (2005) en esta categoría, sólo algunas son extranjeras, como es el caso de IBM, que se presenta en cuarto lugar con 19.94 millones de dólares (Tabla 3).

Tabla 3: Mercado de desarrollo de software por empresa
(En millones de dólares)

	2003	2004	2005
Hildebrando	15.91	41.07	47.75
Softtek	9.72	17.35	33.78
Neoris	19.05	19.83	21.25
IBM	17.66	17.14	19.94
Accenture	3.64	22.92	14.45
EDS	5.27	10.04	13.05
Getronics	-	9.34	8.39
Bursatec	7.66	7.49	7.79
ITS	6.3	6.58	6.57
Qualita	7.67	7.24	6.04
Bearing Point	4.74	4.66	4.61
Adam Technologies	2.87	3.09	3.43
Netropology	1.73	1.86	2.97
Unisys	3.22	2.79	1.93
SyC	3.33	1.91	1.88
DMR	-	-	1.71
Oracle	1.32	1.38	1.49
Avaya	-	0.86	1.29
DynaWare	0.22	0.44	0.9
gedas	0.99	0.74	0.62
Qarta Sistemas	0.14	0.45	0.47
Kernel	1.12	0.31	0.41
Siga	0.3	0.3	0.34
CTI	0.27	0.3	0.29
Capgemini	0.59	0.35	0.26
Migesa	1.79	0.8	0.12
KED	0.09	0.1	0.1
Nextira One	-	-	0.06
SondaPissa	0.54	0.34	0.05
Cima	0.03	0.04	0.04
Nauter	0.02	0.02	0.02
STI	0	-	-
Total	116.18	179.72	202.01

Fuente: Select (2005).

Por otro lado, la tasa de empleo en el ámbito del software creció más rápido que el del resto de la industria de TI en el periodo 2000-2005. Al pasar de 245 000 a 323 000 empleados, con un incremento de 31.9%. De esta forma, en el área especializada del sector la tasa de crecimiento fue del 43.8%, que representó el 20% del empleo total (Mochi & Hualde, 2009) (Tabla 4).

Tabla 4: Empleados en Software (Valor del mercado vs. empleados)

	2000	2001	2002	2003	2004	2005	Crecimiento 2000-2005
EMPLEADOS EN SOFTWARE	244873	239065	269620	279993	288393	322912	31.90%
Subtotal Industria: software en paquete y desarrollo de software	37485	37521	42252	43074	47615	53915	43.80%
Gastos Internos: Desarrollo de software	207389	201544	227368	236919	240778	268997	29.70%

Fuente: Mochi y Hualde (2009).

Ante este breve panorama de la industria del software, queda claro que el sector de esta industria puede ser desarrollado en México es: a) el área de servicios del software, b) el software que se produce *in-house*, y c) el firmware. Si es que las empresas que lo demandan se encuentran dispuestas a subcontratar a otra. Ya que es evidente que existe gran cantidad de software que es producido internamente:

> Es interesante subrayar el contraste entre la alta demanda de la industria manufacturera en tecnologías de la información y su escasa relevancia en la demanda de software empaquetado, lo que se relaciona con la gran importancia del software creado internamente (Mochi & Hualde, 2009, pág. 200).

Sin embargo, se advierte que esta importancia del software creado internamente no se distribuye o se comparte con otras empresas, debido a la existencia de derechos de autor y patentes. Dicho software compone "implementación de soluciones complejas, desde interfases de conectividad hasta software modular o de aplicaciones, generalmente no se reutiliza, se usa sólo para una empresa" (Mochi & Hualde, 2009, págs. 201-202).

En este sentido, Mochi y Hualde (2006, 2009) advierten que si todo este software fuera aprovechado bajo un esquema de los estándares abiertos, sería posible: bajar precios, se reducirían problemas asociados a la piratería en el sector y se reutilizarían recursos ya creados. Los cuales, permitirían a los desarrolladores de software dedicarse a la innovación y no a recrear componentes del software ya existentes, con lo que se malgasta tiempo y dinero.

Tal como se perfila en el panorama internacional, según reportes como el de la UNU-MERIT (2006), los servicios relacionados con el software libre, podrían haber alcanzado 32% de todos los servicios relacionados con las tecnologías de la información para el año 2010, lo que podría significar el 4% del producto interno bruto de Europa. Donde se asegura que en el año 2006 ya sostenía el 29% del software que se desarrollaba para el autoconsumo de las empresas en la Unión Europea (43% en Estados Unidos) y que provee un modelo de desarrollo natural para el sector industrial. Lo cual, resulta coherente, ya que el desarrollo de software doméstico permite realizar soluciones a medida como lo requieren grandes empresas[6], tal es el caso de las que laboran en el sector industrial.

En este contexto, resulta que el software libre ofrece la posibilidad de ser factor para desarrollar los sectores que son susceptibles de crecimiento en la industria del software en México y que, posiblemente le permitan alcanzar el ritmo de crecimiento del resto de América Latina.[7] La pregunta que salta a la vista es: ¿cómo es posible integrar el software libre en la industria mexicana? La existencia de la Asociación Mexicana Empresarial de Software Libre (AMESOL), puede ser un buen lugar para comenzar a buscar estas respuestas,

6 En México son empresas que tienen más de 100 empleados.

7 En Brasil se ha hecho una adopción por decreto presidencial del software libre en las dependencias estatales, sin embargo, estudios hechos por el Comitê Gestor da Internet (CGI, 2010), revelan que en este país las empresas se encuentran realizando la adopción voluntaria del software libre, ya que este estudio reveló que entre más grande era la empresa más penetración existía en el ámbito de sistemas operativos libres. Dado que el software libre requiere que las empresas empleen personal calificado en tecnologías de la información. Así, en 2009, 65% de las grandes empresas usaban sistemas operativos libres, del mismo modo, el 48% de las empresas medianas usaron este tipo de software (lo que significa un incremento de 4 por ciento en relación con el año anterior).

a partir de las prácticas que siguen los empresarios para integrar el software libre en su esquema de negocios.

Ahora que sabemos la capacidad potencial que tiene el software libre para la industria del software en México, pasaremos a realizar una pequeña discusión de cómo es que se ha estudiado el software libre en general. Y posteriormente, se abordará el enfoque que se considera óptimo para observar esta realidad.

b. Los estudios del software libre

Como veremos más a fondo en los capítulos siguientes, el software libre emergió en los años 80 como una distinción del llamado software propietario que prohibía el acceso al código fuente y por lo tanto a la capacidad de estudiarlo, modificarlo y redistribuirlo. Además, esta restricción se encontraba legitimada por la ley de los "derechos de autor" en el derecho continental o *copyright* bajo el esquema anglosajón. El software libre reivindicaba la forma en que se había desarrollado el software desde principios de la década de los 60, como el producto de la cooperación entre programadores y hacía evidente que estas restricciones atentaban con la forma en que el software se había desarrollado (véase: Capítulo I).

Sin embargo, no fue hasta el advenimiento del Internet y al desarrollo de innovaciones técnicas, que esta forma de producción del software fue potencializada. El más vigoroso ejemplo de esto ha sido el desarrollo del *kernel* Linux (véase: Capítulo I y II), que al principio llamó la atención de los desarrolladores que se encontraron absortos ante la manera en que este software tan complejo había sido creado. La cual, puede ser resumida en dos aspectos: la primera es que no existía quien controlase su propiedad intelectual y, la segunda, es que se haya desarrollado a partir de la cooperación de miles de voluntarios (véase: Capítulo II).

Lo anterior, llevó a personas que procedían del campo especializado y eminentemente técnicas a explicarlo desde el ámbito de las motivaciones. Tal es el caso de la publicación de Richard Stallman (2004), fundador del

movimiento del software libre y de la Free Software Foundation, o bien de Eric Steven Raymond (1998, 2000) miembro fundador de la Open Source Initiative (OSI).

Los escritos de estos precursores han atraído la atención de expertos en el áreas de conocimiento de las ciencias sociales, como es el caso de la economía, la ciencia política, la psicología, la antropología y la sociología. Entre estas disciplinas científicas, la economía es la que más ha producido estudios en torno al estudiado del software libre. De este modo, comenzaron a aparecer escritos que resaltaban diversas características del software libre, entre las que podemos distinguir cuatro características principales: a) los que resaltan el papel de las motivaciones, b) aquellos que ponen el acento en conceptualizar el software libre como un bien público *sui generis*, c) tenemos aquellos que hacen énfasis en el aspecto legal que implica este tipo de software para la economía y, por último, d) los que hacen hincapié en las consecuencias que tiene la integración del software libre en el modelo de negocios de las empresas.

En la primer categoría, encontramos los escritos que ponen especial interés en las motivaciones que llevan a los programadores a desarrollar software libre. Particularmente aquel que se mantiene a partir de contribuciones voluntarias, ya que no proporciona retribución monetaria directa como lo haría la venta de una licencia de uso, puesto que existe la opción de dedicar el mismo tiempo a desempeñar otra tarea. Entre este tipo de trabajos podemos contar el de Josh Lerner y Jean Tirole (2000) y el de Guido Hertel, Sven Niedner y Stefanie Herrman (2003).

Ante los escritos anteriores, se desarrollaron trabajos que perfilan la posibilidad de que el software libre no cumpla con los presupuestos de varias teorías económicas como es el caso del que presenta Miquel Vidal (2000). En este sentido, estos trabajos no se preocupan tanto por las motivaciones, sino por el marco de decisiones racionales que llevan a los desarrolladores a crear y colaborar en el mundo del software libre. Entre esta literatura destacan los aportes de Marc A. Smith y Peter Kollock (1999) y David Lancanshire (2001), cuyo principal logro es conceptualizar el software como un bien público (no rival

y no excluyente) con ciertas peculiaridades. Donde al contrario de lo que postula la teoría de la elección racional, el software libre parece beneficiarse de los *free riders*.

Otro aspecto importante que se ha trabajado, es el que se deriva del marco legal que ha inaugurado el software libre, al utilizar las herramientas diseñadas para proteger los derechos de autor o *copyright*, para impedir a todo agente la posibilidad de restringir su acceso. En esta discusión sobresale la aportación de Lawrence Lessig (1999, 2002, 2004), quien ha influenciado trabajos como el de Martin Kretschmer (2003) o el de Paul B. de Laat (2005) entre otros.

Por último, tenemos aquellos que toman en cuenta el papel de las empresas en el software libre y *viceversa*. Por lo que se sugiere que existe un nuevo modelo de negocios en la industria del software encarnado por algunas empresas grandes, frente a la industria de la venta de licencias liderada por Microsoft. Este nuevo modelo de negocios, fundamentalmente se basa en la creación de soluciones a medida, ya que se apoyan en satisfacer las necesidades de un cliente, así como en el servicio de soporte técnico y asesoría. Entre estos estudios encontramos los de Yochai Benkler (2002), Richard E. Hawkins (2002), Joel West (2003), Eric Von Hippel y Georg Von Krogh (2002) y Edgar Buenrostro (2010).

El presente trabajo de investigación parece tener amplia relación con los trabajos anteriores y sobre todo con el de West (2003). Dado que en este último, se abordan casos cualitativos que le permiten construir teoría fundamentada de manera inductiva. Y toma en cuenta la incorporación del modelo de producción del software libre con el de realización de ganancia por parte de las compañías que antes se dedicaban exclusivamente al software propietario, como parte de una estrategia de negocios híbrida. Del mismo modo, observa la presencia dominante de Microsoft como un factor importante que ha impulsado la adopción de software libre en empresas grandes como son: IBM, Apple Inc. y Sun Microsystems. Sin embargo, en su trabajo falta el marco teórico de explicación general, que permite entender estos cambios en una

realidad concreta.

Si bien, el abordaje de este último autor es similar, falta la explicación de cómo es posible la apropiación y la valorización del software en general, así como en las empresas pequeñas en particular. Ya que aunque en AMESOL existen empresas como IBM, esta asociación se encuentra principalmente compuesta por empresas pequeñas (véase: Capítulo IV). Y las cuales, se dedican principalmente al desarrollo de soluciones a medida como subcontratistas, así como en el rubro de servicios y capacitación, áreas que son susceptibles de crecimiento en la industria del software en México, tal como vimos en el apartado anterior.

Como parte de un movimiento general que implica la adopción del software libre, y que se aprecia en la industria del software en general, se puede ubicar la aparición de la AMESOL. Esto incluye empresas a grandes como IBM, Novell, que forjaron sus fortunas a partir de derechos de autor, que les permite vender licencias de uso de su software[8] a los consumidores de computadoras, lo cual, les ha permitido erigirse como parte de las compañías más lucrativas del planeta. Y que ahora, se encuentran integrando software libre en sus estrategias comerciales, es decir, permiten que se estudie, modifique y redistribuya aquello con lo que generan ganancias, donde incluso Microsoft ha incursionado en esta tendencia.[9] Cuando algo así sucede, entonces es momento de asombrarse, ante una realidad que cambia y preguntar tres cosas:

- ¿Cómo es posible que se puedan realizar ganancias a partir de algo que es de libre acceso?
- ¿Cuál es el proceso que ha permitido estos cambios en las empresas como las anteriormente mencionadas?
- ¿Cómo se integran las prácticas del software libre en las empresas de la realidad concreta mexicana?

Por ello, en esta investigación resulta importante la existencia de la AMESOL. Pues concentra empresas que en México usan el software libre y

8 Llamado software propietario.
9 Como referencia rápida véase: Bobbie Johnson (2009) y el Capítulo IV.

permite acceder a ellas.

Ahora bien, para construir el marco de análisis de la realidad concreta mexicana, en este trabajo se han retomado escritos que no forman parte de la corriente dominante en el estudio del software libre. Tal es el caso de la economía política, que nos presenta el trabajo de Steven Weber (2000), así como trabajos que hacen uso de categorías marxistas, como lo hace Sergio Ordóñez (2006; et al. 2008). Del mismo modo, aunque esta investigación tiene harto que ver con el campo de la economía, tiene un abordaje desde las prácticas sociales de los sujetos que utilizan el software libre como estrategia de negocios en una realidad concreta, que se encuentra compuesta por un contexto histórico dado.

Para lograr ser congruente con estas características, se ha optado por un abordaje holístico, que desde una perspectiva histórica, sea capaz de dar un marco explicativo del cómo se han dado las dinámicas que permiten la integración del software libre en esquemas de realización de ganancia. Y de manera particular, en las prácticas sociales que integran las estrategias de negocios de los miembros de la AMESOL.

Por ello, el principio de esta investigación requiere ubicarnos en una realidad cambiante, que rompe los conceptos que la han explicado anteriormente y que requiere que éstos sean resignificados. En este sentido, desde un ámbito general, en este trabajo se retoman las aportaciones hechas por Manuel Castells (2001) y Pekka Himanen (2002), y aunque es sus obras, tocan el tema del software libre de manera incidental, brindan una visión de la realidad en donde se desarrolla este tipo de programas.

Del mismo modo, la concepción de la sociedad de la información, brinda el marco que permite concebir el software como conocimiento, desde los planteamientos de Nonaka (2001). Y que como tal, forma parte de las innovaciones que se realizan en las empresas, como lo apuntan Mónica Casalet (1995, 2000) Jens Nyholm, Lars Normann, Claus frelle-Petersen, Mark Riis, y Peter Torsten (2001), Rullani (2000a, 2000b, 2000c), Yoguel (2008; et al. 2006; et al. 2005) y Robert (2004) (véase: Capítulo I).

Por último, cabe mencionar, que para la presente pesquisa se ha retomado de manera amplia la investigación de Prudencio Óscar Mochi Alemán (2006), quien brinda un marco conceptual y un panorama empírico para entender la industria del software en América Latina y particularmente en México.

Todos los elementos anteriores, deben ser articulados, en un marco congruente de explicación, por lo que a continuación pasamos a detallar los principios epistemológicos, teóricos y metodológicos que brindan el sustento, a una reconstrucción articulada de una realidad que se presenta como compleja y que a la vez, nos habilita el abordaje de las prácticas sociales como un dándose para identificarlas en sus posibilidades.

2. Fundamentos epistémicos, teóricos y metodológicos

Para poder desagregar la realidad, y encontrarnos en posición de responder a las preguntas que se han planteado en el apartado anterior, es preciso abordar el contexto histórico que ha hecho posible el fenómeno del software libre. De esta suerte, dicho contexto permitirá elaborar herramientas conceptuales que nos permitan reconstruir la realidad con un sentido. Este último, es el que hará posible acceder a las prácticas sociales concretas en su dimensión de posibilidad, pues existen en el dándose.

De esta forma, hay que entender no sólo cómo el software es producido, sino cómo se ha realizado su producción en la sociedad capitalista que lo hace posible. Donde si bien, toda producción es social, la apropiación de las ganancias es privada.

Esto nos permitirá poner en contexto y considerar el porqué el software libre es tan disruptivo con respecto a lo que se conocía antes. Así como, el motivo de que se haya configurado como factor de cambio de la industria del software en general y de la manera en que empresas particulares generan ganancias a partir de él.

Como hemos visto en el apartado anterior, existen diversos modos de acercarse a un mismo fenómeno social, que engloban visiones desde distintas

convenciones para dividir el conocimiento humano, como lo son: economía, ciencia política, psicología, antropología o sociología. E incluso, dentro de cada una de estas disciplinas hay multitud de enfoques que permiten observar diversos aspectos de los mismos fenómenos.

Esta investigación, propone realizar un acercamiento sociológico de este fenómeno, que ponga en relación diversas dimensiones de la realidad, que van desde el aspecto histórico hasta expresiones particulares del mismo. Para ello se plantea, en primer lugar, partir de que la producción social es parte de la realidad. Allende el investigador accede a aquella a través de la interpretación, acción que le permite descubrirla. Así, la interpretación sociológica de la realidad, desde un enfoque weberiano, destaca que:

> El propósito de la sociología no es inventar el mundo social (lo es precisamente en el sentido latino del término), sino descubrirlo: conseguir que las realidades sociales sean también categorías sociológicas, ya que descubrir algo es sobre todo conceptualizarlo. Descubrimiento que no es especular, pues de serlo sólo refleja lo dado, lo que es inmediatamente inescrutable, lo que la realidad ofrece como realidad y como apariencia engañosa. Descubrir es, pues, construir conceptualmente la realidad, pero no de manera arbitraria y caprichosa, sino de manera racional de acuerdo con la cultura del discurso crítico, y construirla conforme con la propia realidad, explicando y destruyendo las apariencias engañosas. Construir conceptualmente la realidad es tanto como elaborar un mapa de la misma, mapa que no es la realidad ni su reflejo, pero la representa, interpreta y hace inteligible. Y tal construcción existe siempre: o la hace la ciencia o la hace la ignorancia (Beltrán, 1991, pág. 60).

Como apunta Luis E. Alonso (1998) la realidad social relevante es “una consecuencia *de la perspectiva* que adopta el sujeto que investiga, perspectiva que siempre es selección y construcción” (pág. 21).

Ahora bien, para construir nuestra observación en torno a nuestros conceptos eje, que son la valorización del conocimiento y la realización de ganancia, se tiene que definir qué se ha entendido por estos conceptos en diversos contextos históricos, para de esta manera, estar alerta sobre el tipo de cambios que se han dado, y estar vigilantes de nuevos cambios que se presentan en el devenir histórico.

> Las constantes variaciones en la estructura de la realidad solo pueden captarse desde una mirada a la vez concentrada y atenta, es decir, que por un lado centre su atención

> en unos ciertos elementos, un cierto ámbito de lo real y, por otro, mantenga presente que eso que mira se está moviendo (Andrade, 2007).

En el mismo sentido, esta realidad que se mueve, tiene que ver con sujetos, no con meramente individuos o empresas. Pensar esto así, permite plantear que ellos tienen conocimiento de su realidad y la interpretan, para realizar su autodeterminación. Esto se presenta desde condiciones dadas, en las cuales se realizan prácticas a través de las cuales se enfrentan a su realidad, que a su vez tienen sus propios requerimientos de conocimiento. De esta manera, Hugo Zemelman (1996a), nos señala que la práctica social: "constituye un producto y un ámbito de conflicto de entre fuerzas sociales que luchan por imponer sus proyectos, de acuerdo con la conciencia que tengan sus intereses" (pág.191).

Por lo tanto, para hacer evidente la relación entre nuevas formas de valorización del conocimiento y la realización de ganancias, como una práctica de los sujetos que la desempeñan, es necesario desarrollar una contextualización y una conceptualización sólo para la construcción del objeto de investigación.

Esto permitirá identificar y desagregar estructuras de relaciones, las cuales nos harán posible reconstruir un ámbito problemático delimitado por la especificidad histórica, articulación que encuadra al sujeto y al objeto de investigación en una realidad compleja y en movimiento (Ramos, Argott, & Barrueta, 2004). De esta forma, la construcción de la aprehensión de lo real en movimiento, requiere la inclusión de diversos momentos de la realidad cristalizada, que articulen lo dado (conceptual o empírico) y el dándose, en un campo de observación de relaciones posibles (Zemelman, 1992).

Así, el contexto permite organizar el abordaje del tema de investigación en torno a conceptos eje, que se transforman en conceptos ordenadores de la realidad cuando su contenido permite la preeminencia de lo epistemológico sobre lo lógico-teórico. Para que, en segundo lugar, a través de una reconstrucción fuera de la lógica de la teoría que sirve de matriz, sea posible acceder a la novedad de la realidad que desborda a la teoría (Ramos et al., 2004). La valorización del conocimiento y la forma de realización de ganancia

en la sociedad capitalista, se configuran así en nuestras categorías de análisis, para escudriñar la realidad.

Y esto se puede hacer, teniendo presente que las prácticas sociales se encuentran en espacios y temporalidades delimitadas en una especificidad histórica. Ya que como explica Zemelman (1996b):

> Toda práctica social conecta pasado y futuro en su concreción presente, ya que siempre se mostrará una doble subjetividad: como reconstrucción del pasado (memoria) y como apropiación del futuro, dependiendo la constitución del sujeto de la articulación de ambas (pág. 116).

Hasta aquí, se atiende al cómo ha de lograrse atender a las interrogantes señaladas más arriba. Sin embargo, aún falta por delimitar el qué movimientos de la historia hay que observar, para dar cuenta de los fenómenos que nos interesan. Tal como es el movimiento que ha llevado a producir ganancias a partir de algo que es de libre acceso o bien, observar el devenir que permitió cambios en la las estrategias de negocios en empresas grandes y pequeñas.

A este respecto, Fred Block (2003), nos comenta que Karl Polanyi ve a la sociedad de mercado como moldeada de una manera que se está llevando a cabo por un doble movimiento. Donde existen intereses que son perseguidos en determinados periodos históricos. El propio Polanyi (2007), nos comenta:

> Retornemos de nuevo a lo que hemos denominado el doble movimiento. Dicho movimiento puede ser definido como la acción de dos principios organizadores en el interior de la sociedad, cada uno de los cuales presenta específicos objetivos institucionales, cuenta con el apoyo de fuerzas sociales determinadas y emplea métodos propios (pág. 128).

Si es posible caracterizar la valorización del conocimiento y la realización de ganancia, como estos principios ordenadores al interior de los principios de la producción del software tanto libre como del propietario, entonces podremos acceder a los objetivos que persigue, identificar sus objetivos, dar cuenta de las fuerzas sociales de que disponen y de los métodos que emplean.

Donde lo que existe son prácticas sociales que luchan por imponer sus proyectos con objetivos específicos, de acuerdo a la consciencia de sus intereses. De esta forma, es posible leer la realidad en términos de este doble movimiento histórico que “(...) debe ser leído más como algo tangible que una

metáfora, es de hecho una respuesta política explícita (…)" (Birchfield, 1999, pág. 39 T. del A.).

Por último, otro aspecto importante a tener en cuenta, es que el presente trabajo, también retoma la perspectiva de Joseph Schumpeter (1983), sobre la concepción de la dinámica de la sociedad de mercado. En el cual, según este autor, el capitalismo es un proceso evolutivo y orgánico donde lo que sostiene su reproducción son "los nuevos bienes de consumo, de los nuevos métodos de producción y transporte, de los nuevos mercados, de las nuevas formas de organización industrial que crea la empresa capitalista" (pág. 121). Lo que requiere que continuamente se creen elementos nuevos que destruyen lo que antes era dominante, lo que él denomina la "destrucción creativa" es "el dato de hecho esencial del capitalismo" (pág. 121). Así, toda empresa que quiera sobrevivir debe adaptarse a esta dinámica.

En este marco lo que hay que observar, pues, es el doble movimiento que presenta la dinámica de la destrucción creadora, donde existen intereses con objetivos particulares sujetos a factores que alientan e inhiben la emergencia e instauración dominante de las innovaciones en situaciones dadas históricamente. Y con estas claves de lectura, proceder a abordar la realidad.

Desde estos principios epistemológicos, teóricos y metodológicos, es que se aborda el resto de esta investigación. A saber: que la realidad se encuentra en movimiento, y que la sociedad capitalista en particular se encuentra regida por un doble movimiento que es político, con una dinámica que tiende a la destrucción creativa a nivel de las innovaciones.

Donde nuestros conceptos ordenadores de esta realidad son la valorización del conocimiento y la forma de realización de ganancia. Por lo que resulta necesario resignificar categorías que han sido desbordadas por la realidad siempre cambiante.

De igual modo, es necesario resaltar que las prácticas sociales concretas de los sujetos específicos son relevantes. Dichos sujetos, en general son percibidos como grandes empresas, pero que en esta investigación también se dedica especial atención a las prácticas de los empresarios mexicanos del

software libre. Quienes nos habilitan para apreciar sus acciones como posibilidades de realidades alternativas, puesto que toda práctica es conexión entre pasado y futuro.

3. Delimitación del objeto de estudio

La presente investigación se limita a analizar cómo se realiza la valorización del conocimiento y la manera en que se realiza ganancia a partir del mismo. Atendiendo a la industria del software en general, pero dedicando puntual atención a la integración del software libre en las estrategias de negocio en empresas.

Para ello, se requiere una perspectiva histórica, que debe contrastar la forma en que se realiza la apropiación, valorización del conocimiento y la realización de ganancia en la industria privada del software, con la reconfiguración de las estructuras que exigen las nuevas formas de producción del conocimiento y la manera en la que se realiza la valorización del conocimiento y la reproducción del capital, en particular en las empresas del software libre.

En esta indagación se propone hacer una comparación entre el modelo privado del software, el modelo de las multinacionales y la forma en la que se expresa en el ámbito mexicano, específicamente en la AMESOL.

En este sentido, el constructo llamado “objeto de estudio”, es una serie de coyunturas conceptuales que tienen el objetivo de presentarse como herramientas para pensar la realidad desde el presente. Así, el objeto de estudio, es más bien una actitud que nos permite conceptualizar la realidad del presente en un contexto histórico amplio.

a. Unidades de análisis

En sí, nuestra unidad de análisis es el doble movimiento que presentan las empresas del software a través de sus prácticas. Donde las unidades de observación son las diversas empresas del software que se consideran para esta investigación, a saber: Microsoft, IBM, Red Hat, Novell y las empresas que

conforman la AMESOL.[10]

4. Justificación del objeto de estudio y definición del problema

A través del presente proyecto se busca develar la práctica social presente que realiza la subsunción del software libre en el capitalismo, la cuál es habilitada sólo en un contexto que tiene dimensión histórica de largo tiempo. Así, se parte del principio de que la realidad se construye histórica y socialmente constantemente por las prácticas que proveen sentido. Donde la práctica social de valorización, es decir, de producción del software libre y de apropiación de la ganancia realizada en el mercado, requieren de la comprensión de su sentido en el ámbito histórico, social y subjetivo del presente.

De esta suerte, existe la posibilidad de observar la forma en que se realiza la valorización de un conocimiento que es de libre acceso, por parte de diversos tipos de empresas, donde a su vez, es posible la realización de ganancia privada. Y que resulta potencialmente relevante para la realidad mexicana.

5. Preguntas de investigación

- ¿Cuáles son las implicaciones de las nuevas formas de valorización del conocimiento en el esquema de realización de ganancia capitalista?
- ¿Cómo se realiza la valorización de las nuevas formas de producción del conocimiento por las diversas empresas del software?
- ¿Cómo se expresa de manera concreta estos fenómenos en la realidad mexicana?

6. Objetivo general de investigación

- Analizar las relaciones entre las nuevas formas de valorización del conocimiento en el proceso de producción del software libre y en el esquema de realización de ganancia de diversas empresas que han incorporado el software libre en sus estrategias de negocios, dedicando

[10] IBM, Red Hat y Novell, forman parte de la AMESOL, pero dado su carácter de transnacionales, se ha decidido manejarlas de forma separada.

atención particular al caso de la Asociación Mexicana Empresarial del Software Libre en su contexto histórico.

7. Objetivos Específicos

- Analizar las relaciones potenciales entre los principios del software libre y la forma de valorización del conocimiento.
- Analizar las implicaciones de la producción con los principios del software libre desde el ámbito empresarial para la realización de ganancia.
- Estudiar el modo de producción del conocimiento y su apropiación en el esquema general que presentan diversas empresas del software, tanto transnacionales como mexicanas.
- Develar el doble movimiento de la realidad que permitió el proceso de incorporación de nuevas formas de valorización del conocimiento y su repercusión en las formas de realizar ganancia.

8. Hipótesis

- Los esquemas tradicionales de valorización del conocimiento se transforman como consecuencia de nuevos avances técnicos y nuevas prácticas sociales que delinean nuevas organizaciones productivas.
- La relación entre la valorización del conocimiento y formas de realización de ganancia es dialéctica, por lo que han emergido procesos que expresan la síntesis entre los modelos del software libre y el propietario.

9. Evidencia empírica

Se ofrece una visión de la industria en general a través de: evidencia hemerográfica, declaraciones de diversos empresarios del software, así como activistas del software libre. De igual manera, se realizaron entrevistas semiestructuradas a miembros de la AMESOL y otros empresarios de la industria del software, las cuales, se utilizan para ilustrar una expresión concreta del fenómeno.

En total se practicaron nueve entrevistas en las que participaron diversos especialistas técnicos y empresarios del software. De los cuales no todos forman parte de la AMESOL, y dada la naturaleza técnica de algunas de ellas no pueden utilizarse citas textuales de todas. De esta suerte, sólo se usan siete en el análisis conceptual del capítulo final que atañe a la AMESOL. Sin embargo, todas han sido valiosas y muchos reconocerán sus aportes y comentarios no sólo en el análisis de los datos, sino en el cuerpo de la totalidad de la tesis.

De esta suerte, agradezco por su tiempo y permitirme compartir su conocimiento, al tiempo que hago el reconocimiento apropiado de su participación como informantes calificados, en orden alfabético a: Alfredo A. Ugalde Lozano (Director General de Kuazar Informática, S. A. de C. V.), Arturo Salinas (Director General de SWEN Software & Consulting), Daniel Ceballos Lugo (Presidente de la AMESOL y Director General de Nul Unu S. A. de C. V.), Fernando Vázquez García (Director de Negocios en TI de GSC Asociados), Isaac Lemus Martin (Gerente Comercial de Factor Evolución S. A. de C. V. y Administrador del portal LinuxParaTodos.net), Jesús Eduardo Moreno Sánchez (Director General de TOKONHU de México), José Luis Hernández López (Director de Simbiótica: Seguridad en Redes), Nahím de Anda Martin (Director General de Factor Evolución S. A. de C. V. y del portal LinuxParaTodos.net) y Rafaela Blanca Silva López (Directora General de Acsinet S. A. de C. V.).

Por último, también se ha utilizado como referente, las experiencias que nos compartieron en conferencias, en primer lugar, Richard Stallman en “Copyright vs. comunidad en la era de las redes informáticas”. Pronunciada en el marco del ciclo de conferencias *Software libre & descolonización digital* (Agosto 18, 2009. La Paz, Bolivia). Y en segundo lugar, la que presentaron Pedro Luis Sánchez Armas (Gerente corporativo del área de TI en Peñoles) y Armando Matamoros Gorzo (Líder de proyectos de Investigación de TI en Peñoles) titulada "Integración de Open Source en Peñoles". Pronunciada en el marco de las conferencias de *la asamblea mensual de la AMESOL* (Octubre 29, 2009. D.F., México).

10. Sobre la estructura de los capítulos y el anexo

La presente indagación se sustenta en el principio epistemológico de que la realidad es compleja e indeterminada y que el objeto de estudio, por lo tanto, es una construcción realizada desde una posición social definida por una historicidad particular que traza un doble movimiento.

En el primer capítulo de la tesis se construye una primera aproximación del tema de investigación que orienta la identificación de posibles campos problemáticos. En este sentido, y en primer lugar, se aborda la historicidad general que delinean el contexto del problema, así se aborda el encuadre histórico del software libre. En segunda instancia, se delimita lo que se ha entendido por valorización y desde este panorama, el papel que ha tenido el conocimiento en diversas etapas del capitalismo. Por último se delinean conceptos clave de esta investigación, como lo que ha de entenderse aquí por conocimiento, la forma en que se inserta en el capitalismo actual, como se articula el software libre con esta concepción del conocimiento y como atañe el mismo a las empresas del software.

El segundo capítulo, se detiene a dar una mejor mirada a las bases de organización de producción y técnicas que han hecho posible que el software libre emerja y se potencialice, tras el advenimiento de los desarrollos en las tecnologías de la información, como el que presenta Internet. Pero además, pasa a realizar una conceptualización de su forma de producción y a observar, las diversas motivaciones que tienen diversos sujetos, como individuos o empresas para adoptar el software libre, al tiempo que se señalan las ventajas de utilizar sus prácticas como punto de referencia para esta investigación.

En seguida, el tercer capítulo, nos brinda la caracterización de dos modelos de casos paradigmáticos de la industria del software, en primer lugar el modelo del software privado encarnado por Microsoft, una de las empresas que se puede considerar ejercen monopolios en varios nichos del mercado del software. El segundo modelo de producción es el del software libre, representado por otros gigantes de la industria, como son IBM, Red Hat y Novell. Y se hace especial énfasis en las estrategias que ha seguido cada una

de estas para acceder o retener a los mercados del software.

El cuarto y último capítulo, se dedica a observar en primer lugar, el contexto específico de la realidad mexicana que permitieron el surgimiento de empresas del software en general y del software libre en particular. Así como las condiciones que habilitaron y los apoyos públicos que coadyuvaron a la conformación de la AMESOL como organismo intermedio. También observaremos como se expresa de manera concreta la articulación entre conocimiento de libre acceso y la forma en que se pueden hacer ganancias a partir de él. Y por último, estaremos en posición de describir el doble movimiento que ha seguido la industria del software.

Al final del presente trabajo de investigación, se ha elaborado un glosario de términos teóricos y técnicos centrales. Esto se debe a que esta investigación tiene una aproximación desde las ciencias sociales a productos del campo técnico de las ciencia de la computación, por lo que al encontrarse en la frontera de ambas disciplinas, sea de esperarse que sus lectores no se encuentren familiarizados con la totalidad de términos que se manejan. Ahora bien, dado su carácter de investigación en ciencias sociales, el rigor de los términos es mayor en esta área, y por el contrario, aún los términos técnicos más usados en el ámbito de las ciencias de la computación son escasamente conocidos en el de las ciencias sociales, por lo que éstos tienden a ser explicados en palabras más sencillas.

Capítulo I: Contexto y conceptualización desde el presente de la sociedad capitalista del conocimiento

Con estos tres aspectos nodales, a saber: la complejidad e indeterminación de la realidad, el objeto de estudio como una construcción y la relación dialéctica entre presente y la historia, me propongo empezar a abordar la relevancia actual del software libre, siempre con referencia a su historia particular en un contexto amplio. En este sentido, el área problemática se encuentra comprendida por distintos cambios que han sufrido las relaciones de producción, apropiación y valorización de conocimiento en la sociedad capitalista del presente.

Así, en primer instancia, se aborda una pequeña semblanza de la propia historicidad del software libre y presentar los sujetos que lo ejercen como práctica social. En segunda instancia planteo el contexto histórico desde el cual se piensa aquí al software libre. Y en última instancia, se analiza la construcción del objeto desde un plano teórico, es decir, las conceptualizaciones del conocimiento que nos permiten abordar el software libre como un "bien club" inmaterial o intangible.

La semblanza histórica, tiene por objeto fijar el "sobre qué" se está hablando, ya que se requiere el contexto en el marco que le da sentido a la conceptualización, así se requiere una pequeña síntesis de la historicidad específica que atañe a la aparición del software libre, aunque su eclosión tiene relación con una sociedad en transformación, mucho mayor al propio recorte de la historia del software libre.[11] De esta forma, asumo que nuestra propia realidad se encuentra en movimiento y sólo es posible dar cuenta de lo dado, como momentos conceptuales que nos permiten aprehenderla. Así, también resulta necesario realizar una breve caracterización de los sujetos que ejercen la práctica social de valorización del software libre en la realidad mexicana y en el momento presente, los miembros de la AMESOL.

[11] Como el fin de la guerra fría, la globalización que ha aparecido en los últimos 20 años y la aparición de Internet, o bien procesos de transformación y adaptación que ha sufrido el modo de producción capitalista a lo largo de los años.

Señalar una conceptualización en un contexto histórico, tiene el objetivo de presentar el vínculo que se ha presentado entre conocimiento y valorización del capital, donde se produjeron distintas relaciones hasta el punto en que el conocimiento ha llegado a convertirse en el factor central de la producción. Este marco conceptual, permite pensar al software libre como una expresión concreta del conocimiento, que presenta rasgos nuevos en la relación valorización-conocimiento.

Por último, se requiere la construcción de la valorización del conocimiento como una práctica social, que requiere un constructo conceptual que caracteriza el software libre como un "bien club", que al derivarse de conocimiento formalizado en código, adquiere propiedades de los bienes inmateriales o intangibles.

1. El software libre como una nueva forma de producción y valorización de conocimiento

En el primer apartado se hace referencia al surgimiento del software libre, en el segundo se muestran los matices que ha sufrido la concepción de software libre, con respecto al énfasis que se hace en su aspecto de capacidad técnica sobre sus atributos que lo han definido como libre, y que le han permitido ser aceptado por el ámbito empresarial. Por último, en el tercero, se hace una breve caracterización de los sujetos que ejercen la práctica del software libre en la realidad mexicana y en el momento presente, los miembros de la AMESOL.

a. El software libre emerge

En sus inicios, el código llamado software, se consideraba un subproducto del objetivo principal, la venta de computadoras, donde se estimaba que se realizaba la mayor ganancia posible. Sin embargo, conforme aquellas crecieron en complejidad, el software resultó ser un componente esencial para realizar el valor de uso de las primeras, ya que sin él no eran más que tarjetas de circuitos integrados, adentro de un gabinete.

En el inicio de la industria del hardware, entre 1960 y la primera mitad de

la década de 1970, los programas eran compartidos entre los técnicos que los escribían, en aquel entonces éstos simplemente eran creados para atender problemas específicos y no se buscaba reproducirlos para su venta comercial (Mochi, 2006). E incluso, el software que empezó a ser protegido por licencias y derechos de autor[12], siguió siendo estudiado y redistribuido entre las comunidades que contaban con conocimientos técnicos suficientes.

De esta forma, el software en general tiene una historia en la que figuran de manera paralela el software protegido con derechos de autor y el software que no se encontraba restringido. En este periodo, cuando el software aún no se consolidaba como un producto de consumo de masas, donde éste era sólo utilizado por usuarios con altos conocimientos técnicos, los cuales resultaban suficientes para modificar incluso el software protegido por licencias.

Previamente a que la empresa Microsoft se convirtiera en dominante del mercado orientado a usuarios no expertos, las tensiones entre quienes pugnaban por software protegido por derechos de autor y quienes demandaban que el software se mantuviera como un bien público se incrementaron. Richard Stallman, programador del *Massachusetts Institute of Technology* (MIT), hace un llamado, a través del manifiesto GNU[13], en 1984 para ganar adherentes con el objetivo de construir un sistema operativo[14] que fuera capaz de correr los programas del sistema UNIX sin ser UNIX, y de esta forma no estar sujetos a las licencias que exigía dicho programa.

Stallman (2004) describe su definición de lo que debe permitir el software de la siguiente manera:

1. La libertad de ejecutar el programa sea cual sea el propósito.
2. La libertad para modificar el programa para ajustarlo a tus necesidades. (Para que se trate de una libertad efectiva en la práctica, deberás tener acceso al código fuente, dado que sin él la tarea de incorporar cambios en un programa es extremadamente difícil.)

---

12 Véase por ejemplo: el caso de la carta de Bill Gates (1976) "An Open Letter to Hobbyists*"* a la comunidad de aficionados del *software*, donde hace especial referencia a los aficionados del Massachusetts Institute of Technology Sloan School of Management (MITS)

13 Acrónimo recursivo, que significa GNU No es UNIX.

14 El sistema operativo es, explicado en términos sencillos, un *software* que permite comunicar el *hardware* (parte física) de la computadora con *software* que tiene un fin específico. Por ejemplo: el teclado con un procesador de textos.

3. La libertad de redistribuir copias, ya sea de forma gratuita, ya sea a cambio del pago de un precio.
4. La libertad de distribuir versiones modificadas del programa, de tal forma que la comunidad pueda aprovechar las mejora introducidas (pág. 19).

De esta forma GNU propone, según apunta el propio Stallman (2004), que el software "(...) no es de dominio público. Todos tendrán permiso para modificar y redistribuir GNU, pero a ningún distribuidor se le permitirá restringir su redistribución posterior. Es decir, no estarán permitidas modificaciones propietarias. Quiero asegurarme de que todas las versiones de GNU permanezcan libres" (pág. 37). Aquí, el ideal es que todos los usuarios se beneficiarían del mismo software. Y lo más importante, desaparece la necesidad de saber quién es el dueño del software y de lo que está o no está permitido hacer con él.

Con esta intención, se crea la Free Software Foundation (FSF), y se publica la Genaral Public License (GPL), que utilizan las herramientas legales creadas para proteger la obra intelectual de los autores, la cual ampara su facultad para decidir sobre sus obras. Por ello, la GPL libera el código[15] del software bajo la condición de que toda obra derivada deberá ser también liberada en los mismos o incluso más permisivos términos, y no podrá ser apropiada con la intención de restringir el acceso ella. Así nace la distinción de software libre (*free* software), como aquel software que permite libre acceso al código, modificarlo y redistribuirlo, de forma gratuita o a cambio del pago de un precio.

En los años posteriores al manifiesto GNU, el sistema operativo que tenía por objetivo avanzó en su desarrollo y se crearon casi todos los componentes necesarios para construirlo. Sin embargo, para 1992 el componente central del sistema, el *kernel* y desarrollado bajo el nombre de *Hurd,* aún no estaba listo (Stallman, 2004).

Por otro lado, en 1991, Linus Torvalds había hecho públicos bajo la GPL

[15] El código, al que se quiere tener acceso es el llamado "código fuente", y que tiene la característica de poder ser leído por personas que conozcan el lenguaje de programación que ha empleado el programador original. Este código, posteriormente puede ser traducido en código legible por una computadora (código objeto), a través del uso de programas llamados compiladores (Mochi, 2006).

sus primeros bosquejos de un *kernel* tipo UNIX, en el nuevo medio de comunicación Internet. Tal como él mismo lo describe en un grupo de noticias llamado "comp.os.minix", a quienes utilizaban otro sistema operativo parecido a UNIX llamado Minix[16]:

> Hola a todos allá afuera usando minix -
>
> Me encuentro haciendo un sistema operativo (libre, sólo un pasatiempo, no será nada grande y profesional como gnu) (...) está empezando a estar listo. Me gustaría cualquier retroalimentación sobre cosas que la gente gusta o disgusta de minix (...).(...) Cualquier sugerencia es bienvenida, pero no prometo que serán incorporadas (...) (Torvalds, 1991 T. del A.).

Para 1992, las respuestas de los entusiastas a través de Internet, bajo la dirección de Torvalds, había consolidado el *kernel* Linux[17], el cual se encontraba suficientemente maduro como ser combinado con las demás partes de GNU.

El sistema operativo, llamado ahora GNU/Linux, es un sistema operativo estable y eficiente que compite e incluso supera sistemas operativos comerciales (Wheeler, 2007). Éste se ha mantenido libre y ha crecido a partir de contribuciones producto del trabajo de programadores individuales e incluso corporaciones de la industria del software. Este sistema es usado por la mayoría de servidores de Internet (Netcraft, 2009), y al que grandes compañías como Red Hat y Novell (Wheeler, 2007), brindan servicio, también tiene aceptación en el mercado de supercomputadores (Top500.org, 2008), e incluso empresas como Dell lo han ofrecido preinstalado en computadoras de escritorio y portátiles (Finkle, 2007).

Pekka Himanen (2002) alude a este sistema operativo, como producto de una nueva identidad, propia de los *hackers*[18], que tienen lo necesario para alentar la colaboración, y que requiere un medio técnico para organizar los recursos locales (Castells, 2001). Gracias a los medios de comunicación e información, potenciados por el avance tecnológico mediático.[19] Prudencio

[16] También de carácter restringido en aquella época, aunque fue liberado en el año 2000 bajo la licencia BSD.

[17] El nombre Linux, procede de la combinación del nombre de su creador (Linus) con las letras que componen la palabra UNIX.

[18] Mochi (2004) lo describe como término extendido entre la comunidad científica que significa: "innovación, virtuosismo y estilo ético" (pág. 338 n).

[19] El propio Internet, fue moldeado en su fase primigenia por los hackers, y también experimentó comercialización que lo han hecho accesible y le han brindado nuevos usos que

Mochi (2006) afirma que "[e]l éxito del software libre se debe en su mayor parte a Internet, porque permite que las personas se pongan en contacto entre sí, actuando como un catalizador que acelera el desarrollo y sintetiza el conocimiento en áreas muy específicas" (pág. 65).

Existen muchos otros casos que pueden brindar ejemplos de software libre, a saber: en el rubro de sistemas operativos OpenBSD, FreeBSD, FreeDOS; en el rubro de servidores se encuentra el más usado del mundo, el programa Web Apache; como navegador de Internet, Mozilla Firefox; y para la producción personal, la *suite* ofimática OpenOffice.org, entre otras aplicaciones.

La forma en que se produce el software libre, a través de la cooperación, tiene la particularidad de que todo aquel con suficientes conocimientos técnicos puede aprender de él y utilizarlo incluso para fines que no fue diseñado originalmente. Esta misma característica hace que se considere como "poco amistoso" para las empresas del software, debido al componente que inhibe su apropiación privada y por ello, fue ncesaria la incorporación de ciertos matices a los postulados del software libre para que se lograra su consideración por parte de aquellas.

b. Open source software

En 1998, miembros de la comunidad del software libre[20] crearon el término software de fuente abierta, o bien *open source* software (OSS), el cual postula como objetivo: evitar el término "libre", ya que en inglés el término "*free*", también significa gratis. Así, llama la atención del público consumidor sobre dos asuntos, el primero es que la principal característica del software que producen es que existe acceso al código fuente. Por otro lado, el segundo asunto, es que no todo el software de fuente abierta es proporcionado de manera gratuita.[21]

también han transformado y contravenido su estructura primigenia. Sin embargo, sin la comercialización de Internet no habría sido posible la difusión y crecimiento del software libre, que presenta un caso paradigmático de aquella producción de conocimiento a nivel planetario, como establece Castells (2001).

20 La organización que dio origen a esta escisión se llama la Open Source Initiative (OSI), fundada por Bruce Perens (s.f.) y Eric S. Raymond (1957).

21 Para ilustrar cómo puede ser esto posible, se utiliza el ejemplo de "la receta y el cocinero", donde cualquiera puede tener acceso a la receta de un buen pastel de chocolate, sin embargo, si uno no quiere o no sabe como hacerlo, se puede pagar a un cocinero para que

Esta forma de entender el software, pronto conllevó a una aproximación pragmática de la metodología su propio desarrollo, donde se valora la superioridad técnica del software, sobre los principios éticos de libertad y solidaridad que enarbola el software libre como movimiento social (Stallman, 2007).

A pesar de las diferencias ideológicas que presentan el software libre y el OSS, las licencias que manejan ambos movimientos son similares, de tal forma que el software producido como software libre califica como OSS y *viceversa*.[22]

Al no poderse realizar la apropiación del software, de una manera estándar, en la sociedad el conocimiento capitalista, compañías como IBM, Red Hat y Novell (Wheeler, 2007), utilizan diferentes proyectos de software libre, los combinan y realizan labores de programación necesaria para ofrecer soluciones de software empaquetado como producto final en el mercado. Donde, si bien, el producto que se presenta puede contener elementos de software privado, las mejoras que se realizan al software libre se mantienen abiertas, con lo que las comunidades de software libre obtienen contribuciones al proyecto original.

En este sentido, parecería no haber, después de todo, tensiones o contradicciones en el ámbito del software libre como medio de valorización al capital. Sin embargo, lo que interesa aquí, no es que exista o no apropiación privada del capital, sino cómo ésta se ha construido como una práctica cultural e histórica que niega la propiedad privada de la mercancía como medio de valorización. Donde existe un proceso de subsunción que nació como resistencias culturales a la apropiación privada y al probar ser potentes proceso creativos, dados medios tecnológicos como Internet, estos procesos sufren transformaciones que acaban potenciando nuevas formas de valorización para la realización de ganancia.

Ahora, pasamos a caracterizar a los sujetos que son el foco de atención para la constitución del software libre, como medio de valorización. Quienes se

hornee por uno.

22 En el ámbito empresarial de habla inglesa, se ha preferido utilizar el término OSS al de software libre (Stallman, 2007), debido a las referencias que hace este último nombre a costo cero, sin embargo en otras lenguas que no presentan esta ambigüedad, se hace uso indistinto de estos términos al punto de referirse a este software como F(L)OSS, acrónimo compuesto por *free* (libre) y *open source software.*

han asociado y son sujetos de este estudio, pues son ellos quienes realizan la práctica de realización de ganancia a partir de un bien que no puede ser apropiado de forma privada.

c. Los empresarios del software libre

Los empresarios del software libre, son los sujetos que realizan la valorización de un bien que es de libre acceso. En México varios de ellos se han afiliado en la Asociación Mexicana Empresarial del Software Libre, creada en marzo del 2003, con el objetivo de agrupar y dar representación a la comunidad empresarial del software libre, quienes se encuentran organizados desde la sociedad civil y de esta manera, contar con el reconocimiento de la Secretaría de Economía (Casalet, 2008).

En este ámbito, la AMESOL se asume como parte de la comunidad del software libre y coincide "en muchos de sus objetivos, valores y forma de trabajo" (Ceballos, 2009). Resulta pertinente hacer notar que la existencia de la AMESOL, permite localizar los sujetos que de otra manera se encontrarían dispersos puesto que las tecnologías que les han permitido apropiarse de conocimiento también los hacen dispersos geográficamente y difíciles de ubicar ya que sólo se hacen visibles si se requiere alguno de sus productos o servicios específicos, donde probablemente, para aquel que los consume ni siquiera sepa que ha consumido productos o servicios fruto del uso y/o aplicación del software libre.

Asimismo es importante recalcar, que como parte de la comunidad del software libre, los miembros de la AMESOL tienen acceso a una comunidad, la cual, les brinda la posibilidad de acceder a software libre y modificar el código restringido por ciertos matices de las licencias que utilizan. Donde surge la pregunta ¿cómo hacen los empresarios del software libre para realizar una ganancia a partir de un bien que es de libre acceso?

Para abordar esta pregunta a lo largo de la tesis, es necesario encarar el software libre, no como un producto para el consumo, sino más bien producido por un sistema cultural, que alienta la colaboración y que ha sido potenciado por

un medio técnico, el Internet. Por ello, es necesario pensarlo dependiente de las relaciones históricas y sociales que le han permitido emerger de esta forma concreta.

2. Para pensar el software libre: conceptualización de la relación conocimiento y valorización en la historicidad del capitalismo

En el siglo XIX, Marx reconocía que el capitalismo se basaba en la transformación de la naturaleza, a través de su apropiación por medio de trabajo abstracto, sin embargo, en aquél momento aún no se había reconocido la importancia que tiene el conocimiento que éste se lleve a cabo.

La relevancia del conocimiento en el proceso productivo ha cambiado cualitativamente hasta que al día de hoy sociólogos como Manuel Castells (2002), han señalado que en la sociedad actual "es la acción del conocimiento sobre sí mismo como principal fuente de productividad" (pág. 43). Aquí resulta indispensable reflexionar sobre cómo se ha dado este cambio, lo que permite pensar el software libre como un conjunto histórico específico de relaciones, que le permiten al propio software ser producido y ser valorizado.

Por esta razón, este apartado se divide en tres partes. En la primera se apuntan las raíces de esta línea de pensamiento en la idea marxista del cambio en las relaciones de producción resaltando el papel central que ha tenido el conocimiento. En la segunda, se anotan las características conceptuales que permiten extender los propios términos en los que se había planteado originalmente y ser reformulada en el mismo sentido que lo ha hecho el capitalismo desde la muerte de Marx. Lo cual nos brinda una guía importante para aprehender la realidad compleja en la que emerge el software libre así como la manera particular como se le piensa. Y en la final, desde las bases que nos brindan las dos anteriores nos permite pensar el software libre en el marco de las relaciones de producción que ha requerido para su existencia, como un conocimiento que escapa a la apropiación privada y adquiere la capacidad de valorizar.

a. El conocimiento en las distintas relaciones de producción en su paradigma clásico

En los albores del capitalismo, Karl Marx (2004) consideraba que existían relaciones de producción de tipo artesanal, manufacturera e industrial. Estas conceptualizaciones, pueden ser también entendidas en un proceso histórico por medio del cual se ha logrado escindir conocimiento y proceso productivo.

De esta forma, resulta necesario abordar la forma en que Marx concibe estos momentos conceptuales haciendo énfasis en el papel que juega el conocimiento en cada uno de ellos. Para luego, pasar a conceptualizar la extensión de esta forma de pensar en nuevas realidades.

Así, para aquel autor, la manufactura es donde se realiza el producto uno a uno, con el empleo de herramientas simples, donde el artesano era **dueño** de los medios de producción y **conocía** el cómo se hace y cómo se utiliza aquello que produce en su totalidad.

La colaboración de varios artesanos en el taller de un capitalista, marca la emergencia de la manufactura, acá cobra relevancia las habilidades de cada uno de los trabajadores para el tipo de trabajo que desarrollan en el proceso de producción. Aquí se empiezan a vislumbrar jerarquías entre los trabajadores, sin embargo, el conocimiento sobre la totalidad del proceso de producción aún es susceptible de formar parte del acervo de conocimientos de la totalidad de trabajadores donde sin embargo, no es necesario. En este momento, se puede entender, que la división social del trabajo presente en la manufactura, reduce el umbral de aprendizaje para nuevos miembros del gremio.

Por último, en la fábrica, el uso de la máquina, llevó a la producción de la gran industria, es decir, a la producción de volumen. Lo que favoreció la eficacia productiva de la tecnología, la cual, también resultó eficaz para separar al trabajador del control del proceso productivo, pues la máquina exigía adaptar el movimiento del obrero a la cadencia repetitiva y continua de aquella, y donde el trabajador puede ser intercambiado sin que se detenga la producción de la máquina, como establece Marx (en Gorz & K. Marx, 1977, pág. 33).

En este punto, el conocimiento del obrero sobre el proceso productivo, se

ha segmentado a tal grado, que ya no tiene conocimiento sobre la totalidad de aquello que produce o incluso, del uso que se le puede dar al fruto de su trabajo. Así, el umbral de conocimiento que requería una persona para entrar a laborar a la fabrica, se reducía al mínimo que la tecnología permitía. En este punto se cumple a cabalidad la afirmación de André Gorz (1977) "la historia de la tecnología puede leerse en su conjunto, como la historia de la descalificación de los agentes directos de la producción" (pág. 101).

Hasta aquí, Marx (1984) nos describe tal como había pensado el mundo del siglo XIX, la tecnología era tanto la destreza del trabajador como los medios materiales de la producción[23], en la realización de una actividad orientada a un fin, satisfacer una necesidad, cuyo resultado, es la valorización de la mercancía producida para crear valores de uso.

Ahora bien, podemos decir que la valorización es el resultado de la realización de una actividad orientada a un fin (trabajo), contenido en una mercancía producida para crear valores de uso y cuyo costo ha sido inferior al precio por el que se cambia en el mercado (valor de cambio) (K. Marx & Engels, 1983). Y donde la ganancia (plusvalor o plusvalía) es la diferencia entre el costo de producción de la mercancía y el precio de su cambio en el mercado. Marx (2002) lo describe como el trabajo excedente no retribuido al trabajador, el cual es apropiado por el capital como parte del proceso de valorización.[24]

Para que el conocimiento pasara a ser central en la producción de valor, se tuvieron que dar varios cambios en las relaciones de producción y en la tecnología que permitía comunicarlo y procesarlo.

---

[23] Para Marx (1984): "cosa o conjunto de cosas que el trabajador interpone entre él y el objeto de trabajo, el cual, le sirve de vehículo de su acción sobre dicho objeto" (pág. 43).

[24] Marx (2002) nos dice "(...) en el *proceso de valorización* como *proceso de apropiación* (...) que el plustrabajo sea puesto como plusvalor del capital significa que el obrero no se apropia del producto de su propio trabajo, que ese producto se le presenta como propiedad ajena; a la inversa, que el trabajo ajeno se presenta como propiedad del capital" (pág. 431 cursivas en el original).

b. Cambios de paradigma en la producción: el ascenso del conocimiento como principal fuente de valor

Si continuamos esta línea de pensamiento, más allá de lo que se estableció en la teoría clásica, encontraremos que en el modo de producción taylorista-fordista se llevó la división entre conocimiento y ejecución del trabajo al límite, donde el conocimiento sobre el proceso productivo es despojado de los trabajadores a nivel de su diseño, lo que conlleva una determinada división social del trabajo. Y por consecuencia, se reduce al obrero a adaptarse a la cadencia impuesta por la máquina, al nivel de la cinta transportadora como un componente más del proceso de producción. El taylorismo es aquí de suma importancia, ya que es el que identifica que para vencer la "natural pereza" del obrero es necesario "restituir al patrón el conocimiento del proceso productivo, «horadando» el monopolio del conocimiento sobre los oficios detentado por los trabajadores" (Revelli, 2004).

La producción en serie, trajo aparejada la transformación en mercancía de varios de los aspectos de la vida, ya que también moldeó la aparición del consumo en masa. Esta forma de producción fue característica de Estados Unidos, donde el contexto de la crisis económica de 1929 marcó la introducción de políticas asistencialistas en su interior y la aparición paulatina, pero constante de la delocalización de la producción a nivel internacional, donde se reubica la producción que demanda mano de obra barata hacia otras partes del mundo.

Por otro lado, en Europa, existió respuesta a la producción en masa y el consumo en masa, fenómeno conocido como el surgimiento del obrero–masa (Coriat, 2005) el cual rechazó esta forma de hacer trabajo, lo que demandó el cambio de las relaciones que se practicaban en el taylorismo-fordismo y el surgimiento de lo que ahora se llama postfordismo (Revelli, 2004).

El aspecto central del postfordismo es que cambia el foco que tenía el taylorismo en la individualización de tareas y el reducir al máximo el conocimiento obrero en el proceso productivo. El aspecto central de esta forma

de producción se caracteriza en que el trabajador debe "liberar" su inteligencia de manera consiente y voluntaria en el proceso productivo (Revelli, 2004).

Este nuevo modelo presenta un rompimiento con el modelo anterior, donde ya no se busca la individualización extrema de la asignación de tareas, sino del fomento de la versatilidad del trabajo y de su habilidad para resolver eventuales contingencias. Esto demandó que el trabajador tuviese conocimientos suficientes para resolver problemas, con capacidad de decisión sobre el proceso productivo, lo cual requirió una recomposición de la banda de montaje donde se promoviera la idea de una organización sociotécnica (Coriat, 2005). Este primer aspecto dio pauta a nuevas tendencias que se resumen en la flexibilización de las empresas y la entrada significativa de los procesos automatizados a través de componentes electrónicos.

> A medida que la electrónica ha ido sustituyendo a los simples principios mecánicos, la *composición técnica* y la *composición de valor* de los productos–mercancías *han alterado totalmente las condiciones de producción y valorización del valor mercantil* (Coriat, 2005, pág. 173).

En este momento, se hace evidente que entre el momento álgido de la producción taylorista-fordista y los cambios que ha demandado el posfordismo, se encuentran íntimamente correlacionados con la forma en que se ha llevado a cabo la escisión entre conocimiento y ejecución del trabajo necesario para la producción.

En este proceso histórico, que ha permitido la aparición de nuevos medios técnicos, así como económico-legales que hacen posible la apropiación privada del conocimiento y su valorización. Y también han abierto el camino para la existencia de nuevos medios de comunicación que facultan la transmisión y procesamiento en tiempo real de grandes cantidades de información, así como la comunicación de conocimiento.

La aparición de Internet en el momento en el que se afianzan las políticas neoliberales al rededor del mundo, habilitan el considerar que la revolución de las comunicaciones han admitido la continuación del proyecto postfordista en algunos aspectos y ha reestructurado otros.

Lo que ha denominado Castells (2002) como empresa red, ha sido el

producto de la delocalización de la producción a nivel planetario. Que continúa los principios de flexibilidad de la producción, en un entorno que cambia continuamente. Estas empresas necesitan empleados capacitados para tomar y transmitir decisiones, tanto en la parte de la infraestructura técnica necesaria como en los puestos directivos. Ya que son los principales centros empresariales, "aquellos que generan y consumen una gran parte del tráfico de datos que circula a través de Internet" (Castells, 2001, pág. 269).

En el proceso construcción de la sociedad capitalista del conocimiento, implicó el cambio de las estructuras productivas y tuvo como principales medios las tecnologías de la información que permiten comunicar y procesar conocimiento. Para ello, los códigos que fueron necesarios han sido software, como conocimiento que se ha hecho concreto. Sin embargo, esta reconfiguración tiene repercusiones importantes en la manera en la que se ha concebido la relación conocimiento-valorización del capital y la cual, se esboza a continuación.

c. Reconfiguración en la relación conocimiento-valorización dados nuevos medios técnicos

En este apartado, abordaremos como es que el conocimiento ha permitido la construcción del postfordismo y la empresa red, a través de su integración en el esquema producción y reconfigurándolo en el ciclo de valorización inmaterial como lo han definido Maurizio Lazzarato (1997a). Y que nos da pie para pensar en el conocimiento como parte esencial de la creación de valor.

De esta manera, en la definición del nuevo paradigma productivo como lo es el postfordismo o la empresa red, se ha establecido la posibilidad de una integración del trabajo científico en el trabajo industrial y el de servicios, a través de actividades de investigación, de concepción, de gestión de los recursos humano, donde el conocimiento que "(...) se convierte en una de las principales fuentes de productividad y pasa a través de los ciclos de producción (...)" (Lazzarato, 1997b) de los cuales se puede disponer en el interior de las redes informáticas y telemáticas que se encuentran en posición de dictar el ciclo de

producción y la organización del trabajo.

Ya que se trata del capital social producido por la cooperación comunicativa entre los trabajadores, sintetizada como el "saber hacer obrero" y redefinida como la esencia de la producción. De esta manera, el trabajo de quienes desarrollan la tecnología, sean científicos o tecnólogos, es también "trabajo vivo"[25] y explotado por el capital, la tecnología es otra mercancía en la que se ha objetivado conocimiento.[26]

Por ello, Lazzarato, denomina a un cierto tipo de comunicación como una forma de producción, la cual define como "inmaterial", a la vez que advierte, que la comunicación que es *producción inmaterial*, es la que

> (…) proviene de las actividades intelectuales en lo que atañe al contenido cultural–informativo, el de las actividades manuales para la capacidad de unir creatividad, imaginación y trabajo técnico y manual, el de las actividades empresariales para la capacidad de *management*, de relaciones sociales y de estructuración de la cooperación social de la que forma parte (Lazzarato, 2000).

Por lo cual, el trabajo inmaterial se constituye " (…) en formas inmediatamente colectivas y sólo existe, por así decirlo, en forma de redes y flujos" (Lazzarato, 2000).

En esta lógica podemos entender por qué después de haber pasado por un proceso de "descalificación de los agentes directos de la producción", como

[25] Marx (2004) distingue el "trabajo vivo" del "trabajo muerto", el primero entendido como el tiempo en el que se desempeña trabajo útil para el modo de producción capitalista, en cambio el trabajo muerto, es el trabajo pretérito objetivado, es el mismo capital, que se reinvierte en la producción. Donde distingue entre "trabajo" y "trabajo útil" el primero se refiere a todo gasto de fuerza humana en sentido fisiológico, este trabajo humano abstracto es el que constituye el valor de la mercancía. Sin embargo, por otro lado, él nos menciona que también el trabajo "es gasto de fuerza humana de trabajo en una forma particular y orientada a un fin, y en esta condición de trabajo útil concreto produce valores de uso" (pág. 58).

[26] En este sentido Enrique Dussel (En C. Marx, 1984) nos dice: "La producción o apropiación personal del científico o del tecnólogo han sido convenientemente flexibilizadas por la educación capitalista, a fin de que por el «bien de la humanidad» entreguen sus descubrimientos sin pedir por ello nada o muy poco (nunca la proporción real de la ganancia que producirán al capital). Una vez que el descubrimiento ha sido hecho cuerpo, incorporado, transubstanciado en el capital, éste se transforma en invento. El «descubrimiento» puede dormir por toda la eternidad en los archivos de patentes interesantes pero irreales; los «inventos» son los que pasan a la historia de la tecnología: fueron aquellos que subsumidos por el capital le deben su ser fundado, su realidad, el haber sido mediación de valorización del mismo capital. De esta manera, como robo del descubrimiento, como pago injusto, como asalariados, los tecnólogos son igualmente «trabajo vivo» explotado por el capital (frecuentemente explotados) en su trabajo mismo, en su dignidad, en la posesión de su invento, en el usufructo del mismo" (pág. 55).

sustento de la valorización; ahora para valorizar competitivamente la producción, se requiere de unos agentes productivos calificados, los que deben ser conscientes de su accionar para poder liberar su propia inteligencia en el proceso productivo, pues deben ser capaces de trabajar colectivamente a través de redes y flujos de información, interpretando y emitiendo mensajes, es decir, comunicándose.

El concepto del consumo inmaterial, que propone Lazzarato (1997a) se refiere sobre todo al consumo de información, cuya particularidad es que no se aniquila en el acto de consumo, pues cuando se consume información ésta se multiplica, aunque tampoco se reproduce de una manera mecánica. Y es que la información "(...) no se destruye en el acto de consumo, sino que amplía, transforma, crea el medio ambiente ideológico y cultural del consumidor" (Lazzarato, 2000) por lo que no reproduce la fuerza física de trabajo, sino que transforma a su utilizador. Cada usuario de esa información la interpreta dentro de su propio marco de plausibilidad, el cual tiene referentes biográficos que implican referentes objetivos, subjetivos, temporales y espaciales y que, por lo tanto, actúa en consecuencia a una configuración particular de los mismos (Castells, 2002).[27]

Si concebimos la producción, la distribución y el consumo como una totalidad, donde se realiza la valorización, y entendemos que este proceso se expresa diferencialmente desde que se constituye el capitalismo hasta sus formas más acabadas, donde se ha llegado a producir mediante formas tecnológicas, o como las denomina Lazzarato (2000) "tecnologías de reproducción del saber, del pensamiento, de la imagen, del sonido, del lenguaje" así como mediante nuevas formas de organización y "*managment*", y que han traído consigo un paradigma productivo inédito, nos encontramos frente a este último.

El proceso inmaterial de producción, se puede dividir en el autor, quien

[27] Para explicar esto, podemos pensar a manera de analogía en la enseñanza, la cual, es información consumida de aquel que enseña. Esta información que se desprende de quien enseña no se destruye en el proceso de enseñanza, sino por el contrario, se multiplica. Al final del proceso de enseñanza tanto aquel que enseñaba como quien aprendió poseen, en esencia, el mismo conocimiento.

produce, la reproducción que lo distribuye y la recepción que lo consume. Sin embargo, estos momentos no están separados, la recepción es al mismo tiempo productor y reproductor, pues la producción es social. En el nuevo paradigma productivo se tiene una comunicación organizada y orientada a la producción. Y la reproducción es orientada a la conservación de una rentabilidad, el *público* quien sería el receptor, es también, tendiente a volverse un consumidor/comunicador es decir, un nuevo tipo de cliente.

De esta manera es que Lazzarato nos señala que el problema de la valorización capitalista en este proceso inmaterial, es su legitimidad, dado que –como ya él nos señala– :

> Esta cooperación no puede pre–determinarse en ningún caso por parte de lo económico, pues se trata de la vida misma de la sociedad. Lo "económico" sólo puede apropiarse de las formas y los productos de esta cooperación, normarlos y estandarizarlos. Los elementos creativos, de innovación, están estrechamente ligados a los valores que tan sólo producen formas de vida (Lazzarato, 2000).

Por lo que existe un proceso dialéctico "entre las formas de vida que producen la actividad de los sujetos que las constituyen"(Lazzarato, 2000) y lo económico, que se encarga de gestionar y reglamentar, la actividad inmaterial del trabajo por medio de dispositivos de control, que han creado al público/consumidor, y las herramientas de doble vía en las que se pueden constituir, hablo del potencial dominio sobre las tecnologías de la comunicación y la información.

Estas concepciones sobre la inmaterialidad, tiene características que también se han denominado "intangibles"(Daum, 2003, pág. xi) desde el ámbito económico[28], dada la dificultad que tienen para ser contabilizadas ya que son el producto de las competencias, habilidades y conocimiento de los trabajadores que lo producen.

---

[28] Según nos comentan Enrique O. Luna y Daniel Taritolay (2004): "(...) en la jerga contable, la expresión «bienes intangibles» se emplea con un sentido más restringido que sólo denota a los que reúnen características similares a las de los habitualmente denominados bienes de uso, a saber:
a) se emplean continua y repetitivamente en la actividad del ente;
b) tienen una capacidad de servicio que no se agota ni consume con el primer empleo;
c) mientras están en uso no se transforman en otros bienes ni están destinados a la venta".

Dicho reglamento que impera sobre las nuevas tecnologías, crea la posibilidad legítima de coartar su acceso e impedir su difusión. En el próximo apartado discutiremos la concepciones de conocimiento que nos permite pensar de manera general la forma en que se emplean estas restricciones para impedir su difusión, lograr crear valor, y realizar ganancia privada. Valiéndonos de la concepción del conocimiento y el esquema en el que se inserta en la sociedad capitalista, podemos dar cuenta de la forma concreta en que se caracteriza el software libre, como foco de esta investigación.

3. La apropiación del conocimiento en la sociedad capitalista global y las implicaciones para el software libre

En los comienzos del capitalismo Marx (2002) nos menciona que "[t]oda producción es apropiación de la naturaleza por parte del individuo en el seno y por intermedio de una forma de sociedad determinada. En este sentido, es una tautología decir que la propiedad (la apropiación) es una condición de la producción" (pág. 8). Por otro lado, también señala que toda producción también engendra sus propias instituciones jurídicas, tal es el caso, por ejemplo de la propiedad privada.[29]

El conocimiento, como hemos visto anteriormente, ha sido esencial para la configuración de la producción capitalista en sus diversos estadios de desarrollo. Por lo que en este último apartado, se expone brevemente un recuento sobre la concepción retórica del conocimiento, basados en los principios conceptuales que ha planteado Ikujiro Nonaka, desde una perspectiva dialéctica que contempla la interacción entre el contexto e información, el cual puede ser caracterizado como tácito y explícito. En una segunda instancia, abordamos algunos aspectos de la forma en que se ha llevado a cabo la apropiación privada del conocimiento, y por último se busca examinar una articulación entre el marco histórico y los elementos que hemos abordado antes, que nos permiten pensar el software libre como un

[29] No quiero insinuar que la propiedad privada es única y exclusivamente una característica del capitalismo, pues es claro que existe en otros modos de producción. Sin embargo, al igual que en otros modos de producción también en el capitalismo existen regímenes jurídicos que la protegen y garantizan su perpetuación, *verbigracia* los derechos hereditarios.

conocimiento no apropiable[30] *sui generis* que tiene valor de uso y valor de cambio.

a. Una conceptualización dialéctica del conocimiento

El conocimiento que se ha vuelto determinista, según comenta Enzo Rullani, "tiene la tarea de controlar la naturaleza a través de la técnica y los hombres a través de la jerarquía" (Rullani, 2000b). Los modelos de producción taylorista-fordista, el postfordismo y la empresa red se han valido del cálculo y control técnico, ya que tuvieron éxito en generar grandes ganancias en contextos dados históricamente, al cerrar las posibilidades a las exigencias de la producción. Así, en "el curso de los dos últimos siglos, el conocimiento ha jugado su papel en la objetivación del mundo, adaptando la naturaleza y los hombres a la producción" (Rullani, 2000b). Donde el conocimiento se ha integrado como factor esencial de la producción, parte de las máquinas y el trabajo, es decir, del capital en sí mismo.

Pensar el conocimiento, en este contexto requiere hacerlo de forma tal que se articule con la forma en que se integra en el circuito de valorización que existe en los procesos productivos. La conceptualización de conocimiento que ha presentado Nonaka, se caracteriza por apartarse del principio de "creencia dada por cierta" que rige la concepción occidental, y se apoya en una visión humanista a la que le conciernen diversas dimensiones que pueden ser encuadradas en el ámbito de los procesos productivos.

En este sentido, la experiencia humana es el contexto en el cual la información es conocimiento. Por consiguiente, define el conocimiento como "un proceso dinámico humano de justificar la creencia personal hacia la «verdad»" (Nonaka & Teece, 2001, pág. 15 T. del A.). En este sentido Nonaka hace énfasis en que existen dos tipos de conocimiento: explícito y tácito. El primero se puede

[30] El software libre no es apropiable, más allá del tipo de licencia bajo la que se hace accesible, ya que en el momento en que es apropiado, cuando se cierra el acceso a su código y se prohíbe redistribuir las modificaciones, éste deja de ser libre y se convierte en propietario. Esta distinción aparece por el hecho de que esta acción se realiza en una realidad que tiene tiempo y por lo tanto, momentos distintos. Existen licencias de software libre que permiten hacer esto, como la licencia BSD, sin embargo, ningún software libre impide que sea estudiado, modificado y redistribuido. Bajo la GPL dicha acción es violatoria.

delimitar como aquel que se expresa en un lenguaje formal y sistemático, como pueden ser datos, fórmulas científicas y un lenguaje lógico, etc. Así puede ser procesado, transmitido y guardado de manera relativamente sencilla. Por otro lado, el tácito, es personal y difícil de formalizar, tal es el caso de apreciaciones subjetivas que tienen como base la acción, procedimientos, rutinas, compromisos, ideales, valores y emociones. De esta forma resulta difícil comunicar el conocimiento tácito a otros, pues se trata de un proceso análogo que requiere algún tipo de procesamiento simultáneo (Nonaka & Teece, 2001).

Ahora bien, estos tipos de conocimiento, según explica Enzo Rullani, se encuentran relacionados en lo que denominan el "ciclo cognitivo", que reconoce que el conocimiento es simultáneamente tácito y explícito, el primero que depende de un contexto específico y el segundo separado del contexto y hecho explícito a través de un código. De esta suerte, el ciclo explica como se transforma lo tácito en explícito y viceversa (Rullani, 2000a).

De esta forma, el ciclo se puede desagregar en cuatro pasos:

1) Tácito a tácito: Se difunde por medio de las observaciones empíricas, cuando se opera en equipos que interactúan en un espacio dado. Aquí importa la reflexión que se hace acerca de lo que se hace y de las dinámicas sociales y técnicas que tienen lugar en un contexto específico.
2) Tácito a explícito: Comprende los procesos lingüístico-comunicacionales, que requieren, por un lado, las experiencias tácitas de los individuos y, por otro, se transforma en explícito a través del diálogo e incluso trasciende la interacción cuando se traduce a lenguajes formales.
3) De explícito a explícito: Se caracteriza por la combinación de conocimientos en diferentes contextos y se desarrolla en multitud de ámbitos y situaciones, tal como: la educación, bases de datos, la comunicación social. De hecho, este proceso también involucra agentes que no tienen contacto directo o comparten experiencias que se deriven de algún contexto específico.
4) De explícito a tácito: Se le denomina al proceso que internaliza el conocimiento bajo distintos contextos, sí los conocimientos explícitos se

re-contextualizan y se vuelven tácitos. De esta forma, resulta importante la experimentación por medio de la prueba y el error.

En este ciclo del conocimiento, Nonaka nos presenta como el conocimiento pasa a ser de individual a colectivo, a través de los dos primeros puntos, para luego pasar de colectivo a individual, nuevamente. En este punto Nonaka atribuye a la organización la proyección del círculo cognitivo en el cual se realiza la producción de conocimiento (Nonaka & Teece, 2001).

De esta manera, Nonaka tiene un "modelo simple y elegante para explicar la generación de conocimiento en la firma"(Castells, 2002, pág. 187), que sostiene que las fuentes de innovación se multiplican en las empresas logran establecer puentes entre el conocimiento tácito y explícito, que les permiten mejorar procesos, aunque estos se encuentren ya estandarizados.

Ahora bien, la conceptualización de Nonaka, resulta relevante puesto que, si consideramos que el software es conocimiento que se ha sintetizado en un bien intangible, entonces podemos tener presentes dos situaciones: En primer lugar, el software es el producto de un conocimiento tácito, y en segundo, el software puede ser pensando como conocimiento formalizado o codificado.

Tal como sugiere Ordóñez et al. (2008) el "software constituye conocimiento codificado y plasmado en un programa que permite su inmediata aplicación, posibilitando una articulación directa e interactiva entre el (...) [sector científico educativo] y la producción" (pág. 41 n). Así, el software puede ser considerado como objetivación del conocimiento[31], hecho por los programadores y restringido a través de distintos regímenes jurídicos que se han encargado de legitimar la existencia de la propiedad privada. El contexto presente, que se caracteriza por estar regido por la sociedad capitalista del conocimiento, existe un régimen jurídico particular que se encuentra dedicado a la legitimación de la restricción del acceso al conocimiento.

[31] Esta objetivación sólo podía producirse en organizaciones científico educativas, donde existe gran producción, circulación y acumulación, esto es, el lugar en el que más fácilmente apareció el modo de producir software es en las universidades, donde la circulación del conocimiento es mayor que en las empresas, dada la necesidad que tienen éstas por restringirlo y apropiarse del mismo, tal como veremos en el siguiente capítulo.

Como veremos en los siguientes apartados, trataremos de bosquejar las características de la realidad presente que nos permiten hablar de valorización del software libre y que han permitido realizar ganancia a partir de él. Para luego repasar de manera breve las formas en que se han normado los conocimientos al rededor del software libre y eventualmente caracterizar qué tipo de conocimiento es el que se codifica y como se lo ha entendido.

b. Valorización y apropiación privada del conocimiento

Hablar de la apropiación del conocimiento, requiere hacer referencia a la forma en que se restringe su acceso, hoy en día existe en legislaciones internacionales la concepción de propiedad intelectual, que ha sido ampliamente criticada, puesto que es difícilmente transferible la concepción de la propiedad de algo físico a algo inmaterial o intangible, como lo es el conocimiento (Stallman, 2004; Stallman, Ming, Rendules, & Kembrew, 2007; Lessig, 2004; Rullani, 2000c).

En la sociedad del conocimiento capitalista resulta relevante la existencia del régimen de la propiedad intelectual, que conlleva varios marcos legales que incluyen los derechos de autor o *copyright*, leyes de patentes y marcas registradas (Stobbs, 2000) que proclaman la legitimidad de la apropiación privada del conocimiento. Estos regímenes legales se encuentran orientados a impedir que el producto del intelecto humano particular sea aprovechado comercialmente por otros individuos. Se trata de la legitimidad de la propiedad privada del conocimiento objetivado, aquel conocimiento del que se ha realizado su concreción en un medio físico.[32]

Así, lo que se transmite por Internet, en forma de flujos de información es conocimiento, es producto del intelecto humano. Por lo tanto, es potencialmente apropiable y protegido legítimamente por alguno de los regímenes de propiedad intelectual, lo que hace artificialmente escaso el conocimiento pensado como recurso (Rullani, 2000b).

[32] Toda obra es susceptible de ser protegida bajo los derechos de autor, siempre y cuando se haya plasmado en un objeto físico (WIPO, 2006).

Particularmente, el derecho de autor o *copyright,*[33] que si bien ha existido en diferentes formas desde antes de que exista el capitalismo, es este régimen el que ha brindado sustento jurídico a la apropiación de la producción de la sintetización u objetivación del conocimiento, el software, y que garantiza a su vez la reproducción de esta forma de apropiación. De esta forma, la consolidación de la industria del software ha sido legitimada por el régimen de licencias bajo el marco jurídico del derecho de autor o *copyright d*esde principios de los años 80 (véase: Particularmente el primer apartado del presente capítulo). Y que permite un monopolio a la empresa "poseedora "

> (...) de un determinado conocimiento, al erigir barreras a la entrada a otras empresas por un cierto tiempo para explotar el mismo conocimiento. La empresa "poseedora" del conocimiento se ve entonces en la lógica de vender el mayor número de copias del producto original en el cual inicialmente se objetivó el conocimiento y retardar el proceso de difusión de la innovación (...). Ello no obstante el hecho de que el conocimiento que la empresa "posee" es en realidad resultado de un amplio proceso social acumulativo, en el cual sólo en general el último agregado de conocimiento ha sido realizado por la empresa, pero éste no sería posible sin el proceso social acumulativo previo (Ordóñez et al., 2008, pág. 44 n).

Estas condiciones del capitalismo presente, que se pueden observar en el contexto de la producción del software, han llevado a Rullani (2000b) a proponer que nos encontramos en un capitalismo cognitivo, donde la valorización se realiza de manera diferente al capitalismo a secas, donde la valorización del conocimiento no es estable y presenta multitud de caminos posibles.

Como conocimiento objetivado, el software tiene valor de uso, en tanto que ha sido codificado para cumplir una función, sin embargo, como conocimiento codificado y protegido por derechos de autor, se encuentra doblemente protegido, ya que la forma en que se distribuye (código máquina), es ilegible para el hombre, por lo que se requiere trabajo para descifrarlo, por

[33] Como lo apunta la sección 7.8 del "WIPO Intellectual Property HandbookPolicy, Law and Use" (WIPO, 2006), cuando se menciona: "Hay importantes razones para escoger la protección que brida el *copyright*. En primer lugar, los programas de computadora son básicamente escritos y, bajo el artículo 2(1) de la Convención de Berna, el propósito por el cual un escrito es creado es irrelevante desde el punto de vista que califican como trabajos literarios, si son creaciones intelectuales originales"(pág. 436 T. del A.).

otro lado y en segundo lugar, dadas las leyes del régimen legal, hacer esto tiene altas probabilidades de ser ilegal (véase: Capítulo II). En este escenario, el valor de cambio del propio software se mantendrá constante, a pesar de que el costo de su reproducción sea ínfimo, esto se debe a que su difusión no implica su apropiación ni por los consumidores directos ni por posibles competidores.

No obstante, el software también presenta una particularidad importante, si bien puede encontrarse objetivado en un medio físico, también se encuentra objetivado en un medio que comparte semejanzas con la inmaterialidad o intangibilidad de la información, y se comporta de la misma manera que lo hace el conocimiento.

Esto significa que es posible reproducirlo sin esfuerzo y que habilita para realizar tareas, y cuando se trata de software libre, es conocimiento del que se puede aprender y por lo tanto, apropiarlo como conocimiento. Por esto es que se puede caracterizar al software libre, como un "bien club" al que sólo se puede acceder si se tienen las capacidades cognoscitivas y disposición.

Sin embargo, para que este "bien club" pueda ser valorizado se requiere que se desarrolle como una práctica social, que tiene características particulares. En este punto conviene hacer un paréntesis para explicar el cómo se ha llegado a la concepción de software libre como un "bien club", para ello, es necesario hacer una genealogía del concepto. Observar para qué realidad concreta fue creado y como se utiliza para hacer inteligible esta realidad concreta. Esto se debe a que se trata de un concepto central de esta investigación.

c. Conceptualización del "bien club" llamado software libre

El concepto de "bien club" que se utiliza aquí, se deriva de la propuesta de James M. Buchanan (1965), quien pensó en un modelo que le permitiera salvar el hueco que dejaba tanto la teoría económica neoclásica como la teoría de la economía del bienestar, puesto que ambos casos suponen un régimen de propiedad privada, donde se tienen bienes y servicios que son utilizados o consumidos de manera privada o individual, puesto que él señalaba que

también existían los arreglos que permitían un consumo o propiedad comunal.

En este sentido, también señalaba un hueco en la teoría de los bienes públicos, ya que esos modelos, sólo se aplicaban a casos extremos como si se tratasen de bienes "únicamente públicos" o "únicamente privados". De esta manera, Buchanan (1965) construye su teoría de los "bienes club" como un aporte que cubra todo el espectro de posibilidades en el ámbito de la propiedad-consumo. La cual, se caracteriza por delinear una teoría que modela la cooperación de los miembros, donde existen derechos de propiedad y de consumo que hacen diferencia entre diversos conjuntos de personas.

El trabajo de Buchanan (1965), se apega "en estricto sentido a la organización de miembros o arreglos para compartir donde «la exclusión» es posible" (pág. 13 T. del A.), puesto que si no fuera excluyente se trataría de un bien público. Por otro lado, resalta que la posibilidad de la exclusión es prácticamente universal, dadas las limitaciones físicas de los bienes que él consideraba.

Ya en el siglo XXI y ante la emergencia del comercio mundial y el advenimiento de las tecnologías de la información y la comunicación, que sostienen la afirmación del conocimiento como el factor productivo por excelencia, Marco Ferroni y Ashoka Mody (2004), han hecho particular mención de la existencia de bienes públicos internacionales (BPI), cuya demanda es fortalecida por los nuevos medios masivos de comunicación y la información.

Del mismo modo, sostienen, que en esta sociedad de la información, existen alianzas corporativas que se dan por dos motivos específicos: en primer lugar, para reducir los costos de instaurar y sostener los derechos de propiedad[34]. O bien, en segundo lugar, cuando perciben que puede derivarse un proceso de aprendizaje del cual puede surgir alguna innovación. Donde por otro lado, aparecen los costos cuando estas alianzas forman clubes inaccesibles, lo que conlleva "(...) a la creación y ejercicio de poder de mercado" (Ferroni & Mody, 2004, pág. 14).

En este sentido, Oliver Morrissey, Dirk Willem te Velde y Adrian Hewitt

[34] Estos costos también se les conoce como "costes de transacción" en la teoría económica.

(2004), nos dicen que "[l]a posibilidad de exclusión significa que tal bien no es puramente público, y estos bienes son definidos como bienes club, ya que únicamente los miembros del club reciben beneficios" (pág. 31).

Como vemos pues, en estas concepciones, el "bien club" se encuentra siempre bajo la propiedad de alguien cuyo interés es buscar beneficios a través de tarifas de entrada. Como explica Sandler (2004) :

> La financiación completa de la oferta óptima del bien club depende de la congestión, producción y elementos competitivos. La forma de la función de expulsión es un determinante importante en evaluar si una tarifa puede o no financiar completamente el bien club (pág. 86).

Como podemos observar en los textos de Buchanan (1965), Ferroni y Mody (2004), Morrisey et al. (2004), Sandler (2004), e incluso de Jorge García-Arias (2004), el elemento central de los "bienes club" es el mecanismo de exclusión, donde a los usuarios se les cobra una cuota para poder disfrutarlo.

Ahora bien, si se conceptualiza el software libre como un "bien club", aparecen ciertas interrogantes que han de ser planteadas. Ya que el software no se encuentra supeditado a las barreras físicas como lo plantea Buchanan (1965). El número de veces que el software puede ser reproducido es virtualmente infinito, y puede ser enviado del otro lado del planeta solamente supeditado por las barreras técnicas del ancho de banda (véase: Capítulo II).

En el caso específico del software libre, el derecho de propiedad se encuentra subvertido dada la existencia de las licencias que lo hacen disponible a ser estudiado, modificado y redistribuido. En este sentido, la restricción dados derechos de propiedad es nula en cuanto a la exclusión.

Por otro lado, y en el mismo sentido, no es posible fijar una cuota por acceso a aquello que es libremente estudiado, modificado y redistribuido. Ante todas estas diferencias, pareciera que no es posible pensar al software libre como un "bien club". Sin embargo, si nos fijamos en el común denominador de los autores anteriores, todos resaltan que el distintivo del "bien club" es la existencia de algún mecanismo de exclusión.

Es entonces que la conceptualización del software como conocimiento objetivado resulta importante. Tal como se mencionó en el apartado anterior, el

software libre es susceptible de ser apropiado como conocimiento, en este sentido éste es conocimiento codificado que permite aprenderlo. Por ello, resulta importante el aporte de Gabriel Yoguel et al. (Yoguel et al., 2005), quien desde una posición influida por el enfoque neoschumpeteriano, reconoce que es el conocimiento el que se ha constituido como un factor clave de ventajas competitivas de las redes de valorización definidas por la capacidad de asimilación del conocimiento. De esta manera, el éxito de éstas se sustenta en la generación de capacidades cognitivas que se asocian a la creación de cuasi-rentas estructurales fuertemente influidas por la forma específica que adoptan los regímenes tecnológicos, de gestión del conocimiento y de competencia.

Si hemos definido al software como conocimiento codificado, entonces hay que ver como el conocimiento define la pertenencia a la red, es decir, la pertenencia a una red de conocimiento se encuentra limitada por las barreras que presentan las capacidades cognitivas de la red, lo que Yoguel et al. (2005) denomina un “bien club”.

En la concepción de este autor, al hacerse cada vez más restringido este tipo de bienes, se origina una comunidad epistémica. La cual, consiste en una comunidad de individuos que comparten un lenguaje codificado de tal modo que no es inteligible para quienes no son parte de ella. Otra característica de la comunidad epistémica es que está sujeta a un proceso de circulación y apropiación del conocimiento, de lo que se deriva que la misma comunidad no es libre y se encuentra sujeta a la apropiación privada, tal es el caso de lo que denomina tramas productivas jerárquicas.

Así, la concepción de Yoguel (2005) retoma la importancia de las definiciones anteriores, al proponer que en las redes de conocimiento, el mecanismo de exclusión del “bien club” es la capacidad de los miembros para acceder al conocimiento. De esta forma, el concepto de “bien club” de este autor, pasa de ser aplicado a bienes físicos, como lo hacen los teóricos que se han mencionado antes, y lo aplica a bienes inmateriales o intangibles como lo es el conocimiento.

El caso del software puede ser pensado como conocimiento hecho

concreto u objetivado. Ya que el software se desprende de quien lo escribe y a través de los medios electrónicos, en los que necesariamente debe ser plasmado para cumplir el propósito de su valor de uso, se convierte en información que puede ser reproducida sin esfuerzo y de manera multiplicativa, ya que tanto aquel que lo posee en un primer momento como aquel que recibe la copia tienen el mismo software, pero no es posible apropiarlo como conocimiento si no es posible comprender el por qué funciona.

De esta suerte, el software conceptualizado como conocimiento, es entenderlo como conocimiento que ha sido codificado, y esto lo limita a sólo ser aprovechado por aquellos que tienen la capacidad de decodificar este conocimiento, como parte de una red de conocimiento.

El conocimiento del software libre, según nos explica Verónica Robert (2004) tiene la particularidad de que aunque el código fuente del software libre puede ser descargado de Internet sin ningún problema, no puede ser pensado como un bien público, puesto que: su consumo es no rival dado que "no sufre desgaste alguno con el uso y la copia digital implica su reproducción casi sin límites a costos ínfimos" (pág. 10) y aunque admite que no se puede definir claramente que su consumo sea no excluible, en términos hipotéticos:

> (...) cualquier persona (con acceso a la red) tiene a su disposición el inmenso reservorio de código abierto y libre, las posibilidades de hacer un uso efectivo de este material están limitadas a las personas que tengan la capacidad para manipular, adaptar, compilar y poner en funcionamiento los programas para computadoras que ese código encierra. Evidentemente no todos tenemos las capacidades para hacer esto posible (Robert, 2004, pág. 10).

Así, el uso efectivo del software libre, para modificarlo o adaptarlo, se encuentra limitado por las competencias propias de aquel que quiera hacer uso de él. En este sentido, Robert advierte que es preferible pensar el software libre más que un bien público, como un "bien club". Donde hace falta contar con los conocimientos necesarios para acceder a él.

De esta forma, se explica porque el software libre se encuentra restringido sólo a las capacidades cognitivas de quien quiera tener acceso a la red de conocimientos, y no puede ser caracterizado como un bien puramente

público o puramente excluyente, dado que este conocimiento no se encuentra codificado con el propósito mantenerse no inteligible, sino únicamente con el propósito de su funcionamiento técnico.

Por estos motivos el software libre es conceptualizado como un "bien club", por que cumple con el común denominador de todas las definiciones anteriores de este tipo de bienes, a saber: cuenta con un mecanismo de exclusión. Este mecanismo se basa en el costo de adquirir la capacidad de su aprovechamiento, el cual se traduce en tiempo y esfuerzo por parte de aquel que desea comprender el software libre y apropiarlo como conocimiento.

De esta manera, es como el concepto de "bien club", se adapta a la explicación de una realidad para la que no había sido creado, y esta pequeña genealogía pretende dar cuenta de la evolución que ha tenido el concepto para explicar distintas realidades. Y que la forma en que se entiende aquí es apta para hacer esta parte de la realidad inteligible a la razón.

d. Las empresas del software son intensivas en conocimiento

La industria de software, es concebida como una industria intensiva en conocimiento, ya que depende de recursos humanos calificados y propicia la innovación tecnológica al transformar conocimiento tácito en conocimiento codificado explícito. También es vista como una opción viable para países menos desarrollados, ya que ofrece salarios bien remunerados e incluso con ventajas ecológicas, ya que no contamina. Sin embargo, tal vez el mejor motivo para invertir en esta industria, sea que las inversiones iniciales en infraestructura requieren poco capital financiero (Casalet, 2008).

Ahora bien, una perspectiva que trata las innovaciones como el resultado de un proceso formal e informal, ya que la información, los conocimientos y la habilidad son "necesarios para adquirir, asimilar, usar, adaptar, cambiar y crear tecnologías" (Casalet, 1995, pág. 101) esta es una mirada interactiva de la innovación. El panorama que se puede observar, desde este punto de vista se aleja de la mirada exógena de la innovación, que la considera como ajeno al nivel de factores económicos de cada país así como al nivel de conocimientos y

de habilidades con las que se cuenta, por el contrario, atiende la evolución de las tecnologías a través de su aplicación, donde tienen lugar procesos interactivos. Se trata de una óptica que apela a la complejidad de la multidimensionalidad, donde existe retroalimentación de la innovación, su uso y un ambiente institucional en el que se desarrollan (Casalet, 2000).

Ddesde esta mirada, la innovación no es sólo:

> (...) el resultado de actividades de investigación y desarrollo efectuadas en laboratorios específicos de empresas y centros tecnológicos de investigación (…) [sino], una actividad compleja de adaptación de conocimientos genéricos a conocimientos específicos a partir de competencias desarrolladas por las empresas (Casalet, 2000, pág. 345 a).

Este es un modelo no lineal de innovación (Casalet, 1995; Nyholm et al., 2001), que en el presente, también ha atendido las "transformaciones que han aumentado significativamente la velocidad de procesamiento, almacenamiento y transporte de la información en el sistema productivo y en la sociedad" (Yoguel, 2008, pág. 295). Donde se reconoce que el conocimiento un factor clave que permite transformar insumos en bienes y servicios con mayor valor agregado y generar ventajas competitivas. Donde existe generación, circulación y apropiación de conocimientos, donde se retroalimentan conocimientos tácitos y codificados, allende el conocimiento se valoriza cuando se transforma (Rullani, 2000b; Yoguel, 2008) a partir de procesos formales e informales.

Esta perspectiva resulta muy rica, para ver la forma en que se articula el desarrollo de competencias por empresas capaces de generar procesos de aprendizaje, dada la interacción entre conocimiento tácito y explícito, a la hora de aplicarlos y convertirlos en ventajas competitivas.

Conclusiones

Como se ha mencionado en la introducción, con esta investigación se busca hacer evidentes los rasgos que han hecho posibles nuevas formas de valorización del conocimiento y analizar la relación que guardan al momento de realizar ganancia dentro de una sociedad capitalista. Estos rasgos se han encontrado íntimamente ligados a la aparición de las tecnologías en los

procesos productivos, ahora la producción del conocimiento objetivado se ha convertido en el factor más importante de la producción.

A través del presente capítulo, hemos visto que la historia es compleja, y el recorte que se propone tiene como fundamento presentar en primer lugar la historicidad específica que atañe al software libre. En segunda instancia, se ha mostrado el contexto amplio histórico que presenta la producción capitalista y el ascenso que ha tenido el conocimiento para producir valor y realizar ganancia. En la última parte, se presentó, en primer lugar, que es posible conceptualizar la forma en que se produce el conocimiento que se puede aprovechar en el ámbito capitalista. En segundo lugar, se hizo énfasis en la forma en que se legitima su apropiación privada. Y en tercer lugar, un resumen de las relaciones conceptuales a través de las cuales es posible tener una concepción que hace posible advertir la relevancia que ha presentado el conocimiento en la creación de valor de las mercancías desde una visión amplia.

Así, a partir del primer apartado, es notable que la forma de producción privada del software haya proveido la inspiración del sistema operativo libre GNU/Linux, que permite la libre acceso al código fuente, su modificación y redistribución. Inspiración que se puede pensar en dos sentidos, el primero como derivado de la necesidad de alejarse de la apropiación privada del software, y el segundo como el ejemplo del sistema operativo UNIX. Pues este último, sirvió de estro para su arquitectura.

La producción de software libre, primigeniamente liderada por hackers han demostrado la capacidad creativa de la puesta en práctica de la colaboración a partir del conocimiento inmaterial o intangible sintetizado o codificado en software. Hoy en día, estas prácticas colaborativas, que incluyen a individuos y empresas, como parte de su esquema de valorización del capital, lo que ha requerido la distinción del *open source* software (OSS).

En esta primera aproximación al tema, encontramos en este contexto histórico directo, como parte de una comunidad del software libre, a los miembros de la AMESOL, que los ha posibilitado con código que presenta ciertas matices de restricción, asociables al software libre. Donde surge la

pregunta ¿cómo hacen para realizar ganancia a partir de un bien que es de libre acceso?

En la segunda parte del capítulo, se ha procedido a la contextualización del software libre, a partir de los caminos que presentaron distintas relaciones sociales de valorización. Donde han sido necesarios nuevos avances técnicos, que se resumen en la Internet, que hace posible realizar nuevas formas de transmisión, procesamiento y finalmente, nuevas formas de producción del conocimiento a nivel planetario, al tiempo que habilita la flexibilización del la gestión y la internacionalización del capital, y con el la producción y el comercio.

Ahora bien, en estas redes de producción globales se requieren puestos de trabajo tanto para tomar decisiones como técnicos que mantengan la infraestructura de la red de comunicaciones. En el cual, es imprescindible la habilidad de transformar información en conocimiento útil para crear valor.

De esta forma, en este capítulo se han hecho explícitas las distintas etapas de construcción del capitalismo de acuerdo a características conceptuales del modo de producción, con respecto a tres dimensiones diferentes, que forman parte de la escisión conceptual entre conocimiento y trabajo. Donde se subraya la importancia histórica del conocimiento para la globalización potenciada por el Internet y la forma misma en como se ha construido el capitalismo, que ha llevado de la gestión del taller a la fábrica, a la gran industria y a la delocalización de la producción. En cada una de estas etapas existe un componente esencial, que se resume en la forma como se legitima la escisión entre conocimiento y ejecución del conocimiento. Lo que se ha hecho a través de marcos económico legales.

Tabla 5: Relación conceptual entre conocimiento, trabajo y tecnología

Organización del trabajo	Relación trabajo -conocimiento	Uso característico de la Tecnología	Restricción del conocimiento
Artesanal	El trabajador sabe el propósito de su trabajo y para qué lo hace, en su totalidad.	Tecnologías elementales.	No existe restricción alguna al conocimiento necesario para el trabajo.
Manufactura	Se realiza una cooperación de manera parcelaria y jerárquica.	Selección por habilidades de los trabajadores para utilizar la tecnología.	Se restringe el conocimiento sólo a las tareas asignadas.
Fábrica	El trabajo se encuentra limitado a la exigencia de la máquina.	Uso de máquinas como sujetos principales de la producción.	Existe separación entre el diseño de las máquinas y el conocimiento de uso de la máquina.
Taylorismo-Fordismo	Se reduce a sólo las acciones pensadas por y para el trabajador.	Se reduce al trabajador a la cadencia impuesta por la cinta transportadora.	Dada por la división extrema del trabajo, entre agente de la producción y diseño del proceso.
Fabrica post-fordista	Se conforman unidades cooperativas de producción con capacidad de decisión y conocimiento de una parte significativa o de la totalidad del proceso productivo.	Se utilizan modelos flexibles de producción, que requieren dominio de varios tipos de máquinas y procesos automatizados a través de equipos electrónicos.	El conocimiento es normado y estandarizado, ahí donde se requiere comunicación para consumar la cooperación, se realiza valorización inmaterial.
Empresa Red	Delocalización de la producción a nivel planetario requiere conocimiento de los medios, capacidad de aprehender entornos en constante cambio y capacidad de decisión.	Medios de comunicación electrónica, y de procesamiento de información.	Restricciones jurídicas a la producción y apropiación del conocimiento. Se requiere comunicación para consumar la cooperación, se realiza valorización inmaterial.

Fuente: Elaboración propia.

Esta serie de escisiones del conocimiento y el trabajo puede resumirse en la tabla 5, que aporta un mapa de una conceptualización en distintas etapas históricas y de acuerdo con diferentes características del proceso productivo. En la primera columna, se muestran momentos de la realidad desde la separación conceptual entre el conocimiento y los agentes de la producción en el que se aprecia, en primer lugar, que el conocimiento aunque es elemental, comprende la totalidad del proceso productivo. Posteriormente, el modo de producción artesanal, da paso a la manufactura, que emprende la escisión del conocimiento a través de la división social del trabajo, y luego, es la máquina quien se llega a considerar el sujeto central de la producción. En la segunda parte de arriba hacia abajo, de la primera columna se observa el regreso del conocimiento a los agentes de la producción, hasta constituirse más importante que la tecnología en el proceso de valorización.

En la segunda columna se distingue que la intensidad en el uso de la tecnología, ha aumentado históricamente, hasta constituirse en el sujeto central de la producción y reducir el trabajo a la cadencia de la máquina y de manera aún mas extrema, al ritmo de la cinta transportadora. Sin embargo, en la segunda parte de la columna, encontramos la necesidad de personas con conocimientos capaces de tomar decisiones en el ámbito productivo, aunque la tecnología de la información sea un requisito necesario para su existencia.

Por último, en la tercera columna, se ilustra que aparejado a este proceso, se han incrementado las restricciones al conocimiento que interviene en el proceso de producción, en la primera parte, sobre todo como resultado de la división social del trabajo, y en una segunda instancia, como resultado directo de la coerción producto del marco jurídico que legitima la apropiación del conocimiento, ahí donde se realiza la valorización inmaterial o de bienes intangibles.

Como hemos visto, al principio se presenta la característica de una escisión extrema, que luego es invertida debido a la introducción técnica de los nuevos medios de la comunicación y la información. Estos mismos medios, son los que hacen posible que al ser considerados conceptualmente, sea posible

decir que el trabajo ahora es requerido para sintetizar o codificar conocimiento, el software.

Para pensar la realidad en constante movimiento, se presenta la tabla 6 de conceptos, donde se espera que a partir de los hallazgos de la investigación, la realidad muestre nuevos fenómenos que se dan en este encuadre, como puede ser la propia transformación del modelo de producción del software privado, producto del contexto al que le somete la aparición del propio software libre, que sólo se puede dar cuenta si se observa desde la perspectiva histórica desde la que se está abordando.

TABLA 6: Producción de conocimiento, apropiación de conocimiento y modelos de valorización por modelos de empresas

	Producción de conocimiento	**Modelos de apropiación del conocimiento**	**Modelos de valorización**
Modelo de la empresa capitalista clásica	A través de un modelo cerrado y jerárquico de producción Responde a la retroalimentación del mercado. Se dispone a crear necesidades.	Protección del diseño del proceso y la mercancía físicas, principalmente a partir de patentes.	Se realiza a través de la apropiación de la mercancía física, que contiene trabajo vivo objetivado.
Modelo de la empresa privada del software	A través de un modelo cerrado y jerárquico de producción Responde a la retroalimentación del mercado. Se dispone a crear necesidades.	Registro y licenciamiento de propiedad intelectual : Derechos de Autor y Patentes.	Se realiza a partir del control y apropiación del conocimiento vivo que el programador ha objetivado.
Modelo de la empresa del software libre.	Aprovecha conocimiento sintetizado Adapta y da soporte al software libre.	Licenciamiento del conocimiento no permiten su apropiación privada. Se requieren altos conocimientos técnicos para apropiarse del software.	A partir de la apropiación del trabajo vivo objetivado en productos de adaptación del software y servicios.

Fuente: Elaboración propia.

Esta tabla, nos ayuda a pensar desde los distintos modelos de valorización del capital. Sin embargo, ésta también adelanta lo que se revisará en los siguientes capítulos, como se expresan dichos rubros para las empresas privadas del software, y para las empresas del software libre.

En la tabla 6, se observa en la primer columna distintos modelos de empresas, el primero, “Modelo de la empresa capitalista clásica” se usa como referente histórico de los subsiguientes, “Modelo de la empresa privada del software”, en tercer lugar, el “Modelo de la empresa del software libre”. Por las formas que utilizan para producir conocimiento, la configuración en que se realiza la apropiación del conocimiento y el modelo de valorización que le corresponde.

De este modo tenemos que la empresa privada, se basa en un modelo cerrado y jerárquico de producción que responde a la influencia del mercado o le crea necesidades, y la valorización se realiza a partir de la apropiación de trabajo vivo que se ha objetivado en la mercancía física. En el segundo, la empresa del software privado, la diferencia conceptual más significativa es la apropiación del conocimiento del trabajador. En este cuadro aparecen diferencias significativas con respecto al modelo de la empresa del software libre, aquí se encuadra nuestro ámbito problemático, que tiene como principales características distintas formas de producir conocimiento sintetizado, realiza su protección para que no pueda ser privatizado y en tercer lugar, la apropiación se vuelve a realizar a partir del trabajo.

Como ya hemos señalado más arriba, esta última tabla, abre la puerta al análisis del modo en que se realiza la apropiación y valorización del conocimiento en diferentes modelos y la manera en que ha de realizarse en casos particulares. Este análisis, se llevarán a cabo de manera más detallada en los próximos capítulo de esta investigación y que dará base para abordar el caso empírico que nos incumbe, el esquema de valorización del software libre, en un contexto específico la AMESOL. Donde se presenta de manera concreta la interacción entre conocimientos tácitos y explícitos de manera no lineal.

Donde el conocimiento codificado que es el software se ejerce como un

"bien club", en el que la comunidad no se apropia de manera privada del software, por lo que es posible acceder a él por parte de todo aquel que tenga aptitud, tiempo y los medios. De esta suerte, el software se ha conceptualizado como conocimiento que forma parte de una red de conocimiento a la que potencialmente todos tienen acceso.

Ahora bien, las empresas del software, se caracterizan por transformar el conocimiento tácito en explícito, por ello se les llama intensivas en conocimiento. Por lo que es necesario contar con las habilidades necesarias que permitan adquirir, usar, adaptar, cambiar y crear, conocimiento para contextos específicos en un momento particular. En esta industria existen procesos interactivos, pero se encuentran restringidos por la norma legítima del mismo.

En este sentido, lo que caracteriza a las empresas, que trabajan con el software libre es que su conocimiento es potencialmente asequible mutuamente, no sólo por el código en sí, sino por la forma en que ha de utilizarse, donde las empresas que están tratando de desarrollar el conocimiento específico para una situación particular, pueden compartir sus conocimientos hechos concretos y que pueden ser aplican a otras situaciones particulares. Aquí, la ventaja competitiva no es de quien ha logrado formalizar el conocimiento tácito, sino del que sabe aplicar el conocimiento formalizado, pero genérico para solucionar problemas específicos. Al transformar el conocimientos, este se valoriza, y las posibilidades en las que ésta se concreta es la que exploraremos en las páginas siguientes.

Dicho contexto específico se encuentra inmerso en e el movimiento propio de la realidad, han hecho patentes nuevas formas de producción y valorización del conocimiento y con ello, estrategias que se adaptan a los nuevos escenarios dentro de la sociedad capitalista presente demanda nuevas maneras de realización de ganancia. En este sentido resulta trascendente, pasar al cómo se expresan nuevas formas de producción en el software libre.

Capítulo II: Emerge y se potencializa la forma de producción del software libre

Especificar el cómo se lleva a cabo la producción de lo que en el capítulo anterior empezaremos a clasificar como un “bien club”, remite a varios posibles niveles de análisis. Los cuales pueden referirse a las motivaciones particulares de los productores para: a) producir el software en sí, o bien; b) liberar el producto de su trabajo como software libre. Pero también, a la forma en la que se lleva a cabo su producción y se mantiene a través del tiempo.

En la teoría marxista clásica, el término general de “producción”, que se encuentra indisolublemente ligado a los de: distribución, cambio y consumo, de forma tal que forman un silogismo. Tal como nos explica Marx:

> La producción está determinada por leyes generales de la naturaleza; la distribución resulta de la contingencia social y por ello puede ejercer sobre la producción una acción más o menos estimulante; el cambio se sitúa entre las dos como un movimiento formalmente social, y el acto final del consumo, que es concebido no solamente como término, sino también como objetivo final, se sitúa a decir verdad fuera de la economía, salvo cuando a su vez reacciona sobre el punto de partida e inaugura nuevamente un proceso (K. Marx, 2002, págs. 9-10).

Hoy, la contingencia social de la distribución en el caso del software, se ha visto trastornada dadas la emergencia de las tecnologías de la información y la comunicación, ya que al tratarse de un bien inmaterial o intangible, es posible distribuirlo en tiempo real sin importar distancia, limitado técnicamente por el ancho de banda. También, el consumo del software al igual que en el caso del conocimiento no implica su destrucción, así, su consumo y su aprovechamiento para inaugurar nuevamente un proceso de producción, se encuentran unificados conceptualmente. De esta forma, la distribución social de los conocimientos y del acceso social a las nuevas tecnologías de la información permiten el consumo del software como un “bien club”.

En la primera parte de este capítulo, comenzaremos por hacer una pequeña reseña histórica del cómo surgió, en primer lugar, el modo de hacer software y cómo fue que comenzó a hacerse necesaria la distinción de “libre”

frente a “propietario”.

En segunda instancia ubicaremos históricamente la forma de producción que dio origen a la manera en que se produce el software libre, la cual surge de lo que se conoce como el movimiento de Berkeley.

En tercer lugar, ahondaremos en las formas en que la estructura de organización de la producción, que ya se encontraba de forma embrionaria en aquel movimiento, fue potencializada por las tecnologías electrónicas de la información y la comunicación que emergieron de manera paralela, para ello ejemplificaremos concretamente dos casos que nos ayudan a pensar conceptualmente la forma de hacer software libre.

Por último, abordaremos de manera breve las distintas corrientes que atienden diferentes aspectos que hacen posible producción del software libre, como son las motivaciones y el papel de la estructura de producción en sí y se hará énfasis en las prácticas concretas que permiten la existencia inmaterial o intangible del software libre.

1. Antecedentes: la construcción de una industria

La construcción de la industria del software tiene sentido para esta indagación si se piensa su devenir en el marco de: la restricción económico legal a la difusión del conocimiento codificado como software. Y es relevante, en tanto las tecnologías de la información son el andamiaje técnico que sustenta la sociedad capitalista del conocimiento.

El software es necesario para esta sociedad ya que simplemente es lo único que hace posible el procesamiento de datos y que los equipos (*hardware*[35]) funcionen. Esto, ha hecho que el desarrollo del software se haya encontrado íntimamente ligado al hardware, el cual ha sido caracterizado por Mochi (2006) en cuatro periodos.

El primer periodo, que comprende los años entre 1945 a 1965 es cuando comienza a aparecer un mercado incipiente de software a medida, para equipos

[35] El *hardware* “comprende todos los componentes físicos (mecánicos, magnéticos, eléctricos y electrónicos, incluidos los elementos periféricos) de una computadora o de un sistema de procesadores”(Mochi, 2006, pág. 48).

específicos, integrados a los *mainframes*[36] de IBM. De esta forma, la producción de software como actividad comercial independiente, se inicia con la primera generación de computadoras electrónicas a principios de los años 50 con la Univac I, de Remington Rand e IBM, y junto con ella, nace en 1955 la primera empresa de software, la CUC (Computer Usage Company), es aquí que da inicio la especialización del software para necesidades del cliente.

De la misma forma, en este periodo se forma la asociación de grupos usuarios SHARE, que tal como su nombre traducido al español indica ("compartir"), sus miembros realizaban intercambios de código, lo que le permitiría acumular un catálogo de 300 programas. La consolidación de esta biblioteca de programas, habilitó que se afianzara la industria del software a medida y especializado hacia finales de los años 60.

El segundo periodo, que ubica Mochi (2006) va de 1965 a 1978, se caracteriza por la aparición de las minicomputadoras, que demandaron la constitución de la industria del software básico (sistemas operativos y aplicaciones). También se multiplicaron las compañías en la modalidad de "agencias de servicios". En esta década se favoreció el mercado del software de multi-instalación, ya que fue en este mismo periodo que IBM decidió unificar el sistema operativo de sus productos. Estas decisiones impulsaron el número de empresas de software, que en los Estados Unidos se encontraba entre 1500 y 2000. Este periodo vio erigirse a grandes compañías como Computer Sciences Corporation (CSC), la McDonell Douglas, Oracle y Baan. Otro acontecimiento importante, es que en 1972 aparece el software "*start ups",* introducido por la empresa alemana SAP, que tiene por objetivo ser un "software estandarizado capaz de controlar las distintas fases del proceso de negocios" (Mochi, 2006, pág. 54) conocido como ERP's por *enterprise resource planning.*

A finales del segundo periodo, en 1977, la empresa *Apple,* fundada por Steve Jobs y Stephen Wozniak, introduce al mercado la primera

[36] El *mainframe* es una computadora central que se dedica a tareas de procesamiento de bases de datos que se ingresan de manera externa. Por lo que se usa sobre todo para labores administrativas de grandes empresas o del gobierno.

microcomputadora para uso personal, la Apple II, lo que demandó nuevas características del software.

En el periodo: 1978-1993[37], se estandariza el software empaquetado como una actividad independiente de la producción de hardware (Mochi, 2006). Ya que, es en esta época que se difunde la arquitectura de la PC de IBM como microcomputadora de bajo costo, que se integra como una tecnología más propia de las tareas de trabajo cotidianas.

Junto con la PC se difunde también el sistema operativo MS-DOS[38]. Este sistema fue licenciado a IBM por la empresa Micro-Soft[39], y la cual fue capaz de retener la propiedad intelectual sobre el mismo. Este hecho, abrió la puerta para que se licenciara el mismo sistema operativo entre aquellos que se encontraban produciendo clones de la arquitectura de IBM. Esto fue posible, dado que IBM produjo sus computadores a partir de partes que fabricaban terceros para ella y que retuvieron el derecho de producir lo mismo para otras compañías (Castells, 2002).

Esto último, trajo aparejado dos consecuencias importantes para la industria del software: en primer lugar, se consolidó un sistema operativo dominante, entre los fabricantes de microcomputadoras para el mercado de usuarios no expertos; en segundo lugar y en el mismo sentido, abrió la posibilidad de desarrollar software más rápido y para una diversidad mayor de tareas. Diseñar software que se puede utilizar en computadoras de distinto fabricante y de la misma forma, alentó por un lado la diversidad en la producción de software y también, por otro lado, la entrada al mercado de nuevos productores de hardware. El monopolio[40] del sistema operativo abrió el mercado de las aplicaciones del software para la computadoras personales.

---

37 En este periodo se ubica la aparición del movimiento del *software* libre, el cual se describe en el siguiente apartado del presente capítulo.

38 Este nombre, corresponden a las siglas en inglés de Micro-Soft *disc operative system*, y que sirvió de base para el posterior Microsoft Windows.

39 Ahora llamada Microsoft (Tate, 2000).

40 Por monopolio se entiende una condición no deseada del mercado, en la que existe un sólo vendedor en algún punto de la cadena de producción, lo que anula la libre competencia. El monopolio, permite explotar la renta al máximo de una curva dada de demanda, la cual, también puede ser alterada de manera artificial por la introducción de publicidad que crea nuevas necesidades a los consumidores o bien, nuevas maneras de satisfacer viejas necesidades (Borja, 2002, pág. 959 a).

Aquí se constituye un mercado maduro[41] del software, construido en torno al sistema de Microsoft Windows, el cual se popularizó hasta 1990 en su versión 3 y sus variantes, orientadas a los requerimientos cotidianos. Y fue a través de éste, que también se difunde el conjunto de aplicaciones ofimáticas de la misma compañía llamado Microsoft Office, lo que produjo una estandarización de *facto* entre las computadoras personales para compartir archivos ofimáticos.

En este momento, el comercio de software tiene las características de producción, distribución y consumo de cualquier otra mercancía. No obstante, no se trata de cualquier mercancía, ya que como hemos señalado, se trata de conocimiento que se ha sintetizado en software, el cual es el resultado de habilidades y experiencias. Este conocimiento sintetizado es "lógico no físico, no se fabrica, se desarrolla y no se degrada con el tiempo" (Mochi, 2006, pág. 202).

Así se puede apreciar que en la empresa del software:

> (...) su principal ventaja competitiva reside en la creación y adaptación de conocimiento. Este conocimiento es entendido como una combinación fluida de experiencias, valores, informaciones contextuales y competencias especializadas que dan un cuadro de referencia a la evaluación y asimilación de nuevas experiencias y nuevas informaciones (Mochi, 2006, pág. 224).

A pesar de lo anterior, este conocimiento que se ha sintetizado, ha sido apropiado a través de regímenes legales, en el caso del software predominantemente los derechos de autor y que legitiman un modelo de comercialización[42] el cual requiere producir un medio, ya sea el disco magnético u óptico, después dotarle del contenido y por último distribuirlo hasta el punto de

---

41 El concepto de mercado maduro tiene como referente aquel en que la demanda sólo crece de manera discreta, por lo que no tiene sentido producir en exceso un mismo producto. En este mercado, cobra sentido crear la necesidad de consumir características cosméticas de los productos, donde más que satisfacer una necesidad, se tiene por objetivo retribuir prestigio a su poseedor. En el mercado maduro del software, además de la notoriedad simbólica de este tipo de consumo, existe la coerción de *facto,* que obliga a los consumir de manera continua, para poder utilizar nuevos programas que realizan las mismas funciones que sus propias versiones anteriores (véase: Capítulo III).

42 Dicho modelo, se asemeja al que había seguido la industria discográfica durante al menos 60 años antes de que madurara la industria del *software* e incluso, 500 años antes si se considera la industria del libro (Thompson, 1995).

venta para el consumidor final[43].

De esta suerte, como podemos observar, el software libre es en realidad la forma primigenia en la que se presentó el software y lo fue hasta que se legitimó la práctica de cerrar el código a través del normas económico-legales. Es pues, hasta mediados de los años ochenta cuando se generaliza esta práctica, que se hizo necesaria la emergencia de la distinción "libre". A continuación pasamos a ubicar históricamente la estructura organizacional que permitió la supervivencia de la práctica del software libre, en una realidad donde es posible restringir el conocimiento codificado de manera legítima.

2. Emerge la metodología de hacer software libre en el marco de la sociedad capitalista del conocimiento

En esta segunda sección, observamos el cómo surgió la historicidad de la metodología de producción de hacer software libre entendida en tres partes. La primera, como la organización de la producción en sí, la cual tuvo lugar antes de la proliferación de los tecnologías de comunicación y la información que se resumen en la red de redes: Internet. Esta organización de producción se aborda de forma concreta en el caso del movimiento de Berkeley y el software que éste produjo. La segunda parte atiende la integración de Internet en dicha estructura productiva, y ejemplificada en el caso del desarrollo del *kernel* Linux. Por último se atiende los medios técnicos, además del desarrollo de Internet, que hicieron posible la coordinación de la producción de software libre a nivel planetario.

a. El movimiento de Berkeley y el surgimiento de una nueva metodología de desarrollo del software

El movimiento de Berkeley en el contexto del software libre, no se encuentra relacionado con el papel destacado que jugaran estudiantes de esta universidad en los años sesenta en torno a la libertad de expresión o en oposición a la

[43] Antes del advenimiento del Internet esta era la única forma de distribuir el software, posteriormente y con el crecimiento del ancho de banda y la popularización de la transacciones digitales, fue posible empezar a distribuir y vender software en línea.

guerra de Vietnam. En dicho contexto, se refiere más bien, al papel que jugaron algunos de sus estudiantes en la aparición de nuevas formas de organización para la producción de software.

Como ya mencionamos antes, en los primeros días del software no existía diferencia entre software privado y software libre. La segunda definición surgió después de que se comenzara a proteger el conocimiento codificado bajo derechos de autor o *copyright*. En Berkeley no se inventó el hecho de redistribuir el código libremente, sin embargo en Berkeley, como en otras universidades, era una práctica que había permeado entre los programadores que ahí trabajaban (Leonard, 2000).

En estas condiciones Bill Joy (1954) empezó el desarrollo de una nueva versión del sistema operativo UNIX, llamada Berkeley Software Distribution (BSD), conocida normalmente como BSD Unix. Esto fue posible, ya que las distribuciones de UNIX proporcionadas por los laboratorios Bell en los años 70, incluían el código fuente y permitían que investigadores realizaran modificaciones y extensiones al sistema operativo.

Así, a partir del código base de UNIX, en 1974 se empezó a utilizar una primera BSD, más como una extensión de aquél sistema operativo para las computadoras que funcionaban en su departamento de cómputo, y para 1977, comenzó a distribuirse en otras universidades donde otros hackers llegaban a mejorar el código y mandaban sus propuestas de vuelta a Berkeley.

En 1978, se lanza una segunda versión (2BSD), la cual, a través de mejoras que se realizaron conforme el proyecto creció a través de los años, su arquitectura se separó de UNIX y en su versión 2.9BSD de 1983, se puede considerar como un sistema operativo independiente, más que un conjunto de aplicaciones mejoradas y “parches” (Karels & C. F. Smith, 1998).

A lo anterior hay que añadir que, como tal, la BSD Unix no se consideraba formalmente software libre, sino que sólo se otorgaba a personas o instituciones que tuviesen acceso a una licencia de AT&T UNIX. Aunque en la práctica no se hiciera mucho esfuerzo por verificar la legitimidad de las licencias, y en realidad, cualquiera podía tener acceso al código de UNIX y

mandar mejoras al código base de Berkeley, de esta forma, en la práctica la BSD era lo que hoy conocemos como software libre[44] (Leonard, 2000).

Lo que interesa aquí es, que sin la existencia de Internet fue posible la configuración de una nueva forma de desarrollo en torno al proyecto de la BSD. Como refiere Leonard:

> La contribución más importante de Berkeley no fue el software: fue la manera en que se creó software en Berkeley. Ahí, un pequeño núcleo -nunca más que cuatro personas en un momento dado- coordinaban las contribuciones de una siempre creciente red distribuida considerablemente, la mayoría, programadores voluntarios que realizaban lanzamientos progresivos de software que contaba con mejoras regulares. Al hacerlo, ellos codificaban un templete para lo que hoy es referido como la "metodología del desarrollo del software de fuente abierta". Puesto de manera más simple, los *hackers* de Berkeley montaron un sistema para crear software libre (Leonard, 2000 T. del A.).

Esta organización, se logró consolidar después de que el fundador original, Bill Joy, dejara el desarrollo de BSD[45], y se creara un comité de mando (*core team*). Dicho comité, consistía de un núcleo de programadores clave, que controlaban el acceso al código, y podían conceder derecho de acceso a colaboradores (*committers*) para poder cristalizar sus contribuciones, esto se puede identificar como el primer nivel en la jerarquía en la metodología de desarrollo. En un segundo nivel, se encontraban los colaboradores del proyecto, quienes detentaban aquel derecho a contribuir, que les habían concedido los programadores núcleo. Más allá, en un tercer nivel, se encontraban las comunidades de desarrolladores que proponen cambios, presentan *bugs*[46] y proponen arreglos a los colaboradores. Hoy en día, este mismo tipo de estructura de desarrollo es reproducida de manera muy similar por proyectos de software libre como GNU y Apache (Leonard, 2000).

En la historia de su desarrollo BSD, se ha dividido en ramas diferentes que a través de diversas circunstancias han desaparecido o se han funcionado.

---

44 Esto no pasó desapercibido por AT&T. La cual, a principios de los 90 llevó una confrontación legal con BSD, empero, al final se convirtió en libremente redistribuirle bajo la licencia BSD. Dicha licencia califica como software libre, pero no prohíbe que el producto de las modificaciones al código fuente sean apropiadas (cerradas) de manera privada bajo derechos de autor.

45 Para co-fundar Sun Microsystems.

46 Así se le llama en el lenguaje técnico a errores de programación.

Hoy se pueden reconocer tres ramas de este sistema operativo que son dominantes: FreeBSD, OpenBSD y NetBSD. Las cuales siguen esta misma estructura en su metodología de desarrollo. Por ejemplo, en el año 2000 se puede observar que FreeBSD, cuenta con un núcleo de programadores clave de al rededor de 16 individuos, rodeados de alrededor de 180 *committers* y miles de desarrolladores.

Este tipo de organización pone de manifiesto la jerarquía existente al interior de las comunidades del software libre, *verbi gratia*, en el caso de los desarrolladores, sus propuestas son desechadas en un 90% de los casos y sólo los muy buenos y que se destacan pueden ascender en el escalafón. Pero esta estructura piramidal, también tiene la virtud de la meritocracia y que permite que la calidad del software producido, sobretodo en proyectos grandes, sea ampliamente reconocida (Leonard, 2000).

A pesar de lo anterior, otro proyecto grande y emblemático del software libre, presenta variaciones a la estructura que hemos presentado aquí, y que vale la pena tratar por separado, ya que a pesar de variaciones sutiles, son variaciones sobre puntos nodales de esta indagación.

b. El desarrollo del kernel Linux: la potencialización de una metodología de desarrollo

Si bien el proyecto BSD, fue exitoso en promover una nueva forma de producción de software, su comunidad es pequeña en comparación con la que se ha formado en torno al sistema operativo GNU/Linux. En particular el desarrollo del *kernel* Linux hizo posible completar el proyecto GNU y puso a disposición de todo aquel con conocimiento y disposición necesarios, la posibilidad de hacer uso y modificaciones a un sistema operativo completo, al tiempo que presentó rasgos en su metodología de desarrollo cualitativamente distintos.

De esta forma, resulta relevante para nuestra indagación, notar que el desarrollo de este *kernel*, a principios de los noventas, se presento en el momento en el que existía una demanda en contra de BSD por parte de AT&T

por los derechos de autor, mientras que Linux se encontraba protegido bajo la licencia GPL al igual que GNU, por lo que los desarrolladores que se comprometieron con el proyecto, tenían la certeza de que continuaría siendo libre (Leonard, 2000).

Tal como se mencionó más arriba, Linux nace de la propuesta inicial de Linus Torvalds (véase: Capítulo I) de un *kernel* para lograr desarrollar un sistema operativo parecido a UNIX en la computadora personal, y quien solicitó la colaboración de a través de Internet. Así, este medio de comunicación, fue fundamental para su desarrollo exponencial, sin embargo, al pensar la estructura productiva que lo hizo posible, se encuentran diferencias con respecto a la forma en como se habían realizado otros grandes proyectos de software libre como BSD y GNU.

En primer lugar, se presentó lo que se ha llamado la figura del "dictador benevolente", que "es precisamente lo que se describe así: La autoridad final de la toma de decisiones reside en una persona, de quien se espera que, por la fuerza de su personalidad o experiencia, la use sabiamente" (Fogel, 2007). La figura de Linus Torvalds inspiró este término, ya que en los primeros días de Linux, las decisiones sobre el núcleo se tomaban de una manera que se puede entender como autoritaria.

Empero, hay que resaltar que conforme el proyecto creció, Torvalds dejó de tomar todas las decisiones, y hoy, ni siquiera toma la mayoría de las decisiones, ya que él mismo fue delegando responsabilidades al interior del proyecto, a quienes se conoce como tenientes (*lieutenants*). Personas con conocimientos especializados que son considerados por la comunidad como ideales para cuidar el correcto desarrollo del sistema.

Steven Weber (2000) ha puesto atención en la autoridad que tiene Torvalds, la cual según especifican tiene que ver con su carisma en un sentido weberiano[47], que se encuentra altamente influenciado por su propia historia por

[47] En esta linea de pensamiento, Lerner & Tirole (2000) nos comentan que "algunos atributos resaltan un liderazgo exitoso. Primero, los programadores deben confiar en el liderazgo: que es, ellos deben creer que los objetivos del líder son suficientemente congruentes con los suyos y no contaminados por intereses egocéntricos, comerciales o políticos. Por ejemplo, el liderazgo debe estar dispuesto a aceptar mejoras por ser eficientes aunque no encajen en el diseño original del líder" (pág. 23 T. del A.).

haber sido: a) el iniciador del proyecto, así como; b) se ha ganado el respeto de otros gracias a sus cualidades como programador y; c) capacidad de llegar a consensos, pero también; d) lo respalda el "éxito evolutivo" que ha implicado el proceso de creación de Linux, como un producto de primera calidad, cuya filosofía de desarrollo invita a no intentar arreglar aquello que no está claramente roto.

En su posición de "dictador benevolente", Torvalds es más bien un juez que dirime problemas que requieren atención y más que mandar, participa en las discusiones como cualquier otro desarrollador. Y ejerce su papel, sólo cuando la comunidad ha alcanzado una mayoría considerable y el camino a tomar se percibe como obvio, es entonces cuando el "dictador", indica el camino que se ha de tomar (Weber, 2000; Fogel, 2007; Deek & McHugh, 2007).

Así, hoy la estructura de desarrollo de Linux es jerárquica, con respecto a aquellos que tienen responsabilidades en su desarrollo. Linus Torvalds en la cúspide, le siguen sus tenientes y abajo de ellos se encuentra las personas que se conoce como dueños de áreas (*area owners*) dentro del proyecto.

En este sentido, parecería que se trata un proceso tan jerárquico como el que ha tenido BSD, adempero, dado que Linux fue un vástago de Internet, que potencializó estructuras productivas que ya se habían presentado de forma embrionaria. Por ello, también presenta sutiles y grandes diferencias, que se pueden reducir a dos aspectos, el primero es su masificación, si bien, no se puede determinar cuantas personas exactamente contribuyen[48], las aportaciones que se hacen pueden contarse en el rango de miles.[49]

Este crecimiento exponencial del desarrollo del *kernel* que no podría haber sido posible sin Internet, también se ha debido a ciertos avances técnicos

[48] Solamente de la versión de Linux 2.6.13 liberada el 29 de Agosto de 2005 ala 2.6.25 del 17 de Abril de 2008 han habido 3 810 desarrolladores diferentes y 257 empleadores, entre empresas y fundaciones (Schönitzer, 2008). Si bien no es posible saber cuales de estos empleados han hecho mejoras al código por su cuenta o la empresa les ha pagado para que lo hagan.

[49] El desarrollo del *kernel* Linux paso de 176 250, líneas de código en su versión 1.0 de 1994 a 12 990 041 en su versión de 24 de Febrero de 2010 (Thorsten, 2010). También, podemos mencionar que en un periodo de 4 años pasó de alrededor de 4 000 contribuciones en la versión del 24 de diciembre de 2004 a aproximadamente 11 000 el 9 de octubre de 2008. Lo que implica que se pasó de 60 contribuciones diarias al código base a 120 en promedio (Schönitzer, 2008).

que posibilitaron su desarrollo, este es el segundo aspecto, el desarrollo de software para el desarrollo masivo[50] a través de Internet. Dicho avance, si bien es técnico, implicó la posibilidad de consolidar la estructura social de desarrollo que se ha mencionado anteriormente.

c. Los avances técnicos: la estructura modular y el "*version control system*"

A principios de los años 90, con relativamente pocos participantes, las contribuciones y discusiones relacionadas a diversos proyectos de software libre en general y de Linux en particular, se hacían a través de listas de correo electrónico, grupos de noticias y a través de descargar archivos desde algún servidor (Deek & McHugh, 2007). Cuando proyectos, como el del *kernel* Linux crecieron, surgió el problema de coordinar las contribuciones de los tenientes, el código de los dueños de área y aquellos parches propuestos por desarrolladores comunes.

De esta manera, para acoplar los diversos esfuerzos hubo dos desarrollos técnicos que consiguieron reducir la complejidad de estas interacciones y agilizar el proceso en Linux. En primer lugar, el paso de una estructura monolítica del *kernel*, con la que se había iniciado el proyecto, a una estructura que permite cargar módulos de forma dinámica, en su versión 2.0 y en segundo lugar, la capacidad que proporcionaron nuevos programas de supervisión de versiones del software.

La modularidad de la estructura del *kernel* significó dos cosas. La primera es que el sistema se encuentra compuesto por piezas que, según el nivel jerárquico en que se encuentran, éstas pueden ser puestas en funcionamiento durante su ejecución si los módulos en los niveles internos del kernel de los que dependen ya se encuentran en funcionamiento. La segunda implicación, es que permitió a los programadores trabajar en diferentes módulos

[50] La conectividad masiva resulta también un problema, que si bien potencia el desarrollo del software, también tiene la posibilidad de afectar a miles de personas con sólo un *e-mail* sobre la más trivial discusión que pueda haber en algún proyecto de software que tenga un tamaño considerable (Fogel, 2007; Deek & McHugh, 2007).

sin interferirse mutuamente y añadir funcionalidades sin reestructurar el *kernel* (Ilustración 2).

En la ilustración, se representa la manera en que se constituye el kernel Linux, como se puede llegara a distinguir, cada recuadro representa un módulo, y cada módulo puede llegar a contener varios componentes. El diagrama se encuentra dividido en cuatro, el centro y la primer capa del centro hacia afuera, es donde se encuentra el código más genérico que representan el 25% del código. La tercer capa, nos presenta el relativo al procesador específico, que equivale a 25% del total. Por último, la capa exterior, nos presenta los *drivers*[51]*,* que comunican el *kernel* con el hardware específico (Curtis, 2010).[52]

El propósito la lámina, es brindar una idea de la manera en que este sistema ha sido conformado, ya que cada cuadro en el interior de cada capa representa un módulo, es decir, una aportación cuyo desarrollo es independiente del resto, pero que puede requerir de algunas otras para poder ejecutarse. Esto es, que son independientes en su dimensión como innovaciones y en su modificación posterior, pero pueden ser interdependientes en la dimensión de su ejecución como valor de uso.

El tipo de software que permitió que esta integración pudiera realizarse con mayor agilidad se llama *version control system* o *concurrent versioning system,* y permite llevar un historial sobre quienes y qué cambios ha realizado cada colaborador al código. El *Concurrent Versions System* (CVS), es el software que permitió, en los primeros años de desarrollo de Linux, que las diversas versiones de sus módulos se mantuvieran en orden y en funcionamiento. Esta herramienta fue creada en 1986 y para 1989 era los suficientemente madura para que comenzara a ser utilizada por desarrolladores de software. De este modo, en 1993 este instrumento estuvo listo para cuando la comunidad que se configuró en torno a Linux comenzó a crecer.[53]

51 Programas que proveen compatibilidad de sistemas operativos con hardware.

52 Como se aprecia, 75% del código del kernel Linux se dedica controlar componentes de hardware específicos.

53 Si bien, este sistema no fue adoptado del todo por Torvalds, ya que prefería que se realizaran revisiones “manuales” del mismo. Pese a ello, al pasar de los años y conforme maduraron estas tecnologías, aceptó integrar una herramienta similar de manera completa a través del uso del programa BitKeeper y después, cuando el acceso a éste último se detuvo, el propio Torvalds desarrolló una nueva aplicación llamada *git* (Torvalds, 2005). La licencia

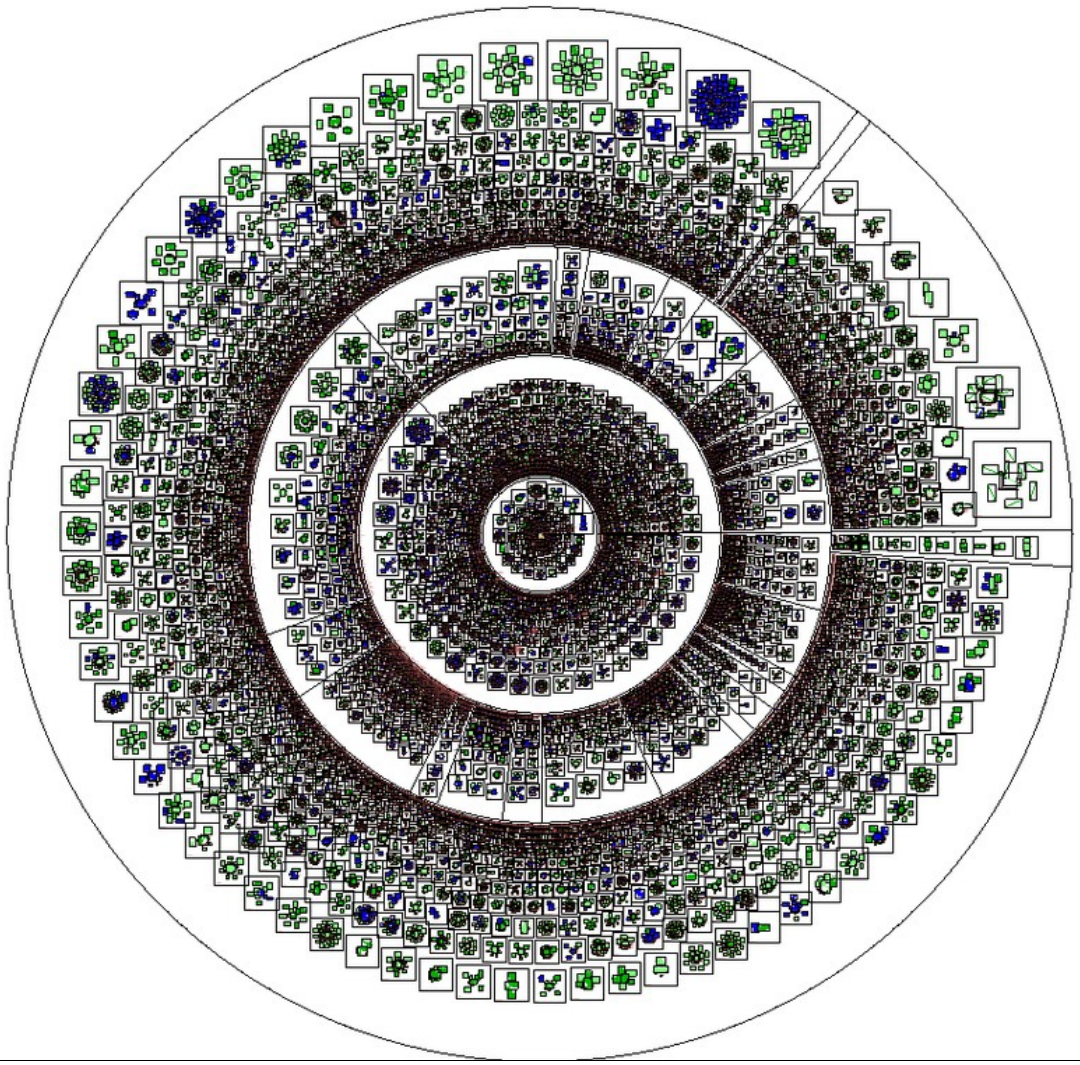

Ilustración 2: Representación gráfica del kernel Linux v2.6.11.8 "Woozy Beaver"

Fuente: Oregon State Linux Users Group. En Curtis (2010)

de BitKeeper es propietaria, y a pesar de ello su dueño y también desarrollador de Linux, Larry McVoy (1962), permitieron a diversos proyectos de software libre, entre ellos el *kernel* Linux utilizarlo sin costo, lo que por otro lado, permitió usar este arreglo como estrategia para incrementar la base de usuarios de su programa (Deek & McHugh, 2007).

El CVS se caracteriza por ser un sistema centralizado de desarrollo, ya que coloca el software que se está perfeccionando en lo que se conoce como repositorio, es decir, en un servidor que lo hace accesible para modificación sólo a desarrolladores con permisos de administración. Lo que dificultaba poder hacer pruebas y modificaciones sobre el código original, ya que obedecían a la lógica de servidores conectados a clientes. De manera concomitante, este aspecto técnico reforzó el carácter jerárquico de la metodología de desarrollo del software libre.

Para atender la verticalidad técnica de desarrollo, surgieron otras versiones de este tipo de software. Por consiguiente, para ofrecer la capacidad de realizar desarrollo alternativos al código surgió Subversion, como sucesor de CVS, que introdujo la posibilidad de administrar diversos arboles que comprenden diferentes ramificaciones de código como objetos de primer orden, y los cambios que se hacen al código son “cosas” que se derivan (al comparar ramificaciones adyacentes).[54] Lo anterior, permitió a los programadores, ser capaces de “experimentar” con el código, aunque también obedece a un esquema centralizado que impone la existencia de un servidor principal.

Por otro lado, para atender las necesidades de proyectos que sean capaces de realizar un desarrollo descentralizado, como es el caso de Linux (Wheeler, 2005) se desarrollaron programas como Bitkeeper, GNU Arch o *git*, que invierten la lógica en la que trabajan CVS o Subversion, ya que manejan los cambios como objetos de primer orden, donde el repositorio es una colección de modificaciones y los árboles son los que se derivan al integrar ciertos conjuntos de aquellas modificaciones. Dicho esquema, permite trabajar asincrónica e independientemente, al hacer actualizaciones en repositorios locales al acoplar cambios de otros desarrolladores y lograr coordinar modificaciones hechas de forma descentralizada, o sea, no se requiere un servidor central.

[54] “En Subversion, un número global de revisiones 'N' nombra a un árbol en el repositorio: como la forma en que se observa al repositorio tras N número de modificaciones. Pero también es el nombre de un cambio implícito que ha tenido lugar al comparar el árbol N con el árbol N-1, tú puedes derivar el parche exacto que ha sido empleado” (Subversion, 2010 T. del A.).

Al final de este apartado resulta pertinente la pregunta, ¿el proceso de desarrollo del software libre es centralizado o descentralizado? La respuesta conforme a lo que hemos visto hasta ahora, es que el desarrollo del software libre ha cambiado conforme han aparecido las nuevas tecnologías de la información y la comunicación. Mientras que en un principio, proyectos como GNU o BSD eran centralizados, en el sentido de que se encontraban, como los describe Raymond (1998), encerrados a piedra y lodo (para mayor detalle en este tema véase: Capítulo III). En cambio, el modelo del software libre que emergió de la mano de Internet, se encuentra altamente abierto a las posibilidades de contribuciones provenientes de diversos desarrolladores independientes o empresas.

Empero, adentro de estas metodologías de desarrollo, también existe la posibilidad de distinguir jerarquías en dos niveles, en aquellos que tienen acceso a los repositorios y hacer efectivas sus modificaciones al código, ya sean los que son parte de algún comité o aquellos que se han ganado la confianza de la comunidad y/o el iniciador del proyecto en el que se han involucrado. El segundo nivel, se refiere al centro gravitatorio alrededor del cuál gira la comunidad de un proyecto: ¿es en torno a una persona, un comité o una empresa? O bien ¿es en torno al proyecto en sí mismo?

Por ejemplo, BSD y Apache no dependen de un centro personificado en una figura identificada como el iniciador del sistema, ya que es el comité que se ha formado el que continua su dirección. Ahora bien, Linux tiene la efigie de Torvalds, quien algún día, y como todo mortal, ya no estará más en el proyecto. No obstante, dada la estructura de este *kernel*, grandes decisiones sobre el futuro del proyecto dependen de aquel que toma el código y la capacidad para desarrollarlo, no de quien es el “dictador benevolente”.

En el presente, grandes compañías como Red Hat, Novell e IBM realizan las contribuciones más voluminosas al código del *kernel* Linux, es decir, en mayor porcentaje que cualquier otro desarrollador de manera individual. Verbigracia: en la versión 2.6.27 del *kernel*, las compañías antes mencionadas lo hicieron con 9.20%, 5.60% y 5.40%, respectivamente (Schönitzer, 2008).

También empresas como Google, lo han utilizado para crear nuevos sistemas operativos como es el caso de Android[55] o bien el Chrome OS[56]. A pesar de ello, se percibe que la organización de producción que se ha conformado para producir software libre ha sido capaz de mantenerse e incluso, de crecer continuamente. Ya que ha logrado atraer apoyo del gran capital[57], proveniente de diversas fuentes y que incluso en algunos planos son rivales.

Por último, podemos señalar que tal como el taylorismo, cambió la forma de organización del trabajo al interior de la fábrica al identificar y posteriormente, extraer el conocimiento al obrero, de la misma forma, el modo de construcción de software libre, ha cambiado la estructura de organización de producción del software, al permitir acceso al conocimiento.

De igual manera, el fordismo que se sirvió de elementos técnicos como la linea de montaje, el medio técnico de Internet y los programas de control de versiones han servido para potenciar esta nueva forma de organización de producción inmaterial o intangible. Sin embargo, conviene destacar cuales son los rasgos centrales de esta forma de producción de manera sistemática, aunque de alguna forma ya se han tocado más arriba de manera tanto implícita como explícita.

3. Conceptualización de la producción: para distinguir motivaciones de prácticas sociales

El propósito de hacer esta conceptualización es mostrar la relevancia histórica que tiene la restricción del conocimiento para la estructura de la organización social y técnica de producción del software libre. Asimismo, hacer evidentes sus

---

55 Sistema operativo para dispositivos móviles, tales como Celulares, Hand Helds, NetBooks y Pads.

56 Sistema operativo que se basa en una aplicación tipo Navegador de Internet y sólo se dedica a utilizar servicios accesibles a través de la red de redes.

57 El código que se aporta directamente al *kernel* por grandes empresas no es el único indicador del apoyo de las mismas. Así, por ejemplo IBM forma parte de los miembros *Platinum* de la fundación Linux, mientras que empresas como Google y Novell son miembros *Gold* y empresas como Red Hat o Toshiba tienen membresía *Silver* (TLF, 2010). Lo cual, como podemos observar no se refleja proporcionalmente en el código que aportan, sino en el dinero que donan para que la fundación promueva, proteja y estandarice Linux al proveer recursos unificados y servicios necesarios para que el software de fuente abierta pueda contender con sus semejantes privativos.

implicaciones para realizar ganancia en el marco de la sociedad capitalista del conocimiento.

En consecuencia, y a partir del contexto que se ha descrito anteriormente, se puede pasar a enumerar las particularidades que tiene la producción del software libre, las cuales podemos dividir en su estructura organizacional, pero también pensándola como prácticas que le dan sentido a la estructura.

a. Sistematización de la estructura de producción

Este apartado tiene la intención de presentar de manera sintética, aspectos que se han visto de manera dispersa en la realidad del contexto histórico concreto, que sirven de encuadre a este problema de investigación. El trabajo de Edgar Buenrostro (2010), ha mostrado de manera sintética estos aspectos, y aunque aquel trabajo ha servido de guía conceptual para este apartado, aquí se trata de hacer con referencia a los ejemplos concretos que se han presentado anteriormente, el desarrollo de la BSD y Linux.

De esta forma, en cuanto a la estructura de desarrollo en comunidades, las lecciones que nos han enseñado estos proyectos es que se encuentran organizadas jerárquicamente:

- En la cúspide se encuentran aquellos que dirigen el proyecto y tienen un mapa de desarrollo, como es el caso de BSD con su equipo núcleo (*core team*). O bien, tienen facultad de incidencia en el camino a seguir en el desarrollo del código, como es el caso de los *hackers* del kernel en Linux. En ambos casos, en la cúspide se encuentran aquellos que tienen acceso de escritura en los repositorios donde se encuentra el código base del proyecto y tienen la facultad de conceder derechos de escritura a colaboradores que consideran calificados para hacerlo. Ya sea a la totalidad del código que comprende el proyecto, un módulo o áreas del mismo.
- En un segundo nivel, los miembros de la comunidad que tienen facultad de escritura al código base (*commiters*), y que tienen capacidad de incidir

en el desarrollo a alguna parte del código ya sea a un módulo o a áreas del mismo. También pueden servir como puente entre colaboradores que han desarrollado código pero no tienen acceso al código para realizar su contribución.

- En un tercer nivel, tenemos a desarrolladores *(developers),* que empiezan agregando "parches" pequeños a alguna parte del código, generalmente a través de algún *commiter* especializado. También pueden caracterizarse por agregar documentación al software o bien, realizar traducciones, *scripts*[58] o extensiones del mismo.
- Por último, en la base de la pirámide se encuentran los usuarios del software, que normalmente colaboran en los foros en línea. En dichos sitios pueden dirigirse para buscar o prestar ayuda, reportan *bugs* y participar en discusiones sobre las características o aplicaciones que hace falta desarrollar.

También, podemos distinguir, que la organización del software libre tiene pues aspectos descentralizados y centralizados.

- En primer lugar, es posible distinguir que en el presente es geográficamente descentralizada, pues los desarrolladores hacen sus aportaciones en línea a través de Internet.
- La estructura organizacional del proyecto "oficial", si bien tiene jerarquía, existe el potencial que cualquiera que haga méritos suficientes ante la comunidad, pueda acceder y ascender en la misma, lo que hace posible una entrada descentralizada tanto desde el punto de vista geográfico como de estructuras institucionales. Si bien, el que desarrolla código puede ser debido a que una empresa le paga, pero también puede ser porque lo encuentra interesante (este aspecto será tratado con más detalle en el siguiente apartado). Apache, BSD y Linux, han aprovechado ventajas de la arquitectura que les permite un desarrollo apartado, descentralizado de *facto,* sea que exista una linea definida por una mesa directiva como lo es en el caso del BSD, o sea, un proceso más laxo

[58] En informática, se refiere a un conjunto de instrucciones que permiten automatizar una tarea.

como es el caso de Linux.

A estos aspectos centralizados y descentralizados del software libre, la posibilidad técnica del desarrollo en módulos ha posibilitado:

- Desarrollar código, independientemente de las modificaciones que se hagan en otros módulos. En este sentido, el desarrollo modular es descentralizado.
- Coordinación, a través de algún tipo de software de control, lo que permite unificar y hacer una versión “oficial” del código exista o no un servidor central.
- Los módulos pueden ser utilizados en otros desarrollos, siempre que se observe la licencia bajo la cuál han sido liberados.

Tomando en cuenta este último punto, se puede pensar, que el resultado final del software no es centralizado, dada la consecuencia social que tiene su desarrollo. Sin dejar de considerar, que después de haber sido producido el código, en la sociedad capitalista del conocimiento, el costo de su apropiación tiene dos dimensiones.

La primera, es su apropiación como cosa, es decir, su posesión directamente proporcional al de su reproducción y redistribución que es ínfimo. Y por otro lado, su segunda dimensión, es el costo de su apropiación como conocimiento, ya que se requieren conocimientos especializados en programación.

Este último costo es más alto, ya que simple y sencillamente, la distribución social del conocimiento se encuentra condicionada por la división social del trabajo y el tiempo socialmente necesario para aprenderlo, por la misma razón que es posible pensar el software libre como un “bien club”.

A esto se añade el interés que han tenido los desarrolladores por involucrarse en proyectos de software libre, sus motivaciones, que han atraído el interés de varios estudiosos del software libre (Hertel et al., 2003; Lerner & Tirole, 2000; Raymond, 1998; Stallman, 2004) y que les llevan a proponer varios esquemas de explicación de su existencia basados en ellas, sin embargo, más relevante para esta investigación es la existencia en sí de su

práctica, que le ha conferido permanencia y posicionamiento como forma de producción de software que ha sido subsumida en la estructura de valorización del gran capital y que al mismo tiempo funge como parte de la infraestructura de la sociedad del conocimiento capitalista.

b. Las motivaciones

Las motivaciones se han caracterizado en relaciones económicas de tipo costo-beneficio, sin embargo, se ha reconocido que guardan relación con aspectos técnicos, políticos, psicológicos o sociales, y dependiendo del enfoque de los diversos estudios se asigna más atención a cada uno de ellos.

Los aspectos que se han tomado de las motivaciones, pueden dividirse en dos ramos, el primero con respecto a la forma en que presentan como particulares de los sujetos que se dedican a producir software libre, y las segundas en torno a las razones que tiene una empresa para acoger su desarrollo.

Las primeras, son las que han distinguido primero, ya que fueron las que dieron origen al fenómeno del software libre, pues se apreciaba como un pasatiempo de entusiastas, ya que se hacían contribuciones de manera primordialmente voluntaria, en cambio, las segundas se comienzan a distinguir cuando se empieza a observar la inversión de capitales de riesgo y de grandes empresas en el sector.

De esta forma, las motivaciones personales que se han identificado y trabajado en la literatura pueden resumirse como sigue, sin que el orden en el que se presentan sea indicador de su relevancia, dado que se presentan de manera arbitraria:

- Necesidad de satisfacer una necesidad. Se ha marcado que una de las motivaciones para crear, modificar y extender software libre es para crear valor de uso, el cual pueda ser aplicado de manera inmediata y sin pasar por el mercado (Ordóñez et al., 2008). Crear software bajo un esquema libre tiene la ventaja de potencialmente aprovechar mejoras que haga la

comunidad al software, y el cual ya cubre una necesidad propia.[59]

- Aprovechar software que ya está hecho y requiera nulo o poco desarrollo. Esto significa que otra motivación potencial, par usar software libre, es utilizar conocimiento ya codificado que requiera poco trabajo para adaptarlo a las necesidades del usuario final. En este caso, se prefiere utilizar tecnologías ya maduras y con amplia gama de usuarios que funcionan como *testers* para dar cuenta de *bugs*, por lo que puede llegar a considerarse como un estándar robusto y confiable.
- Pago directo al desarrollador. Una motivación que también se ha considerado en la literatura, es el pago directo para crear software libre que puede proporcionar una empresa a un programador. Asignar dicha tarea puede estar relacionado con resolver algún *bug* y cuya solución, finalmente acaba por beneficiar a la comunidad del software libre (Lerner & Tirole, 2000).
- Programar software libre es un *hobby.* Otra referencia al por qué se han involucrado en software libre dada de manera personal por los programadores es que programar es relajante y les distrae del estrés cotidiano.
- Reconocimiento de colegas. La motivación psicológica que implican los elogios y muestras de aprecio de la comunidad del software libre.
- Notoriedad en el mercado laboral. Invertir en el desarrollo de software libre también puede significar la posibilidad de un trabajo mejor a la que tiene actualmente, al ser reconocidas sus habilidades como desarrollador.
- Satisfacción por hacer algo “noble”. Entre las explicaciones dadas por participantes del software libre, existen quienes afirma dedicarse a ello porque implica oponerse a formas egoístas de producir software, en tanto no comparten el código que producen (Weber, 2000).[60]

[59] Esto puede ser hecho por un programador particular que tiene una necesidad inmediata, pero también es una estrategia que puede ser válida para una empresa de cualquier tamaño.

[60] Es notable la existencia en este rubro la existencia de oposición específicamente a Microsoft, que es la empresa que tuvo mayor éxito de todas al establecer sus sistemas operativos protegidos bajo el esquema de licencias como dominadores de *facto* del mercado de

Todas las motivaciones anteriores, pueden ser asignadas a los programadores a título personal, sin embargo las empresas también pueden tener motivaciones de carácter estratégico para involucrarse en el desarrollo del software libre. Particularmente estas motivaciones han llamado el interés de los economistas, específicamente desde el enfoque del problema del *free riding* (Lancanshire, 2001; M. A. Smith & Kollock, 1999; Vidal, 2000), al considerar el software como bien público (no rival y no excluyente), con la particularidad de que se beneficia del *free riding* ya que "se observa un crecimiento de las aportaciones de los particulares en proyectos de este tipo de software" (Buenrostro, 2010, pág. 3). Como pueden ser los que se nombran a continuación:

- Estandarización abierta. En el caso del software libre, Linux ofrece la posibilidad de eliminar el problema de dedicar recursos de empresas grandes a mantener un sistema operativo propio, como es el caso de distintas versiones de UNIX desarrolladas por diversas compañías por ejemplo: el Solaris de Sun Microsystems, el AIX de IBM, el HP-UX de HP, o el Irix de Silicon Graphics (Berlecon, 2002) y cuyas aplicaciones pueden ser convertidas a Linux de manera sencilla[61], donde además, tienen la posibilidad de incidir en su desarrollo y tienen la confianza de que no será controlado por ninguna otra compañía gracias a su licencia GPL.[62] Esta característica abierta de la estandarización, a su vez permite a estas compañías en algún momento dado la apropiación del mapa de ruta (*roadmap*), del desarrollo del software que en algún momento dado sea descontinuado por sus *mantainers* o bien, crear su propio proyecto a partir del código ya desarrollado.
- Bajo costo. Éste es cero por concepto de licencias en el caso del software libre, empero, existe posibilidad de coste por el servicio de soporte, aunque éste, puede ser más bajo que el costo total por

computadoras personales.

61 Ya que Linux es un *kernel* parecido a UNIX (*UNIX-like operating system*) (Berlecon, 2002).

62 Sin embargo, está por verse cuanta influencia puede tener una compañía del tamaño de IBM en el futuro desarrollo de Linux. Aunque como hemos visto, existen diferentes maneras en las que se puede influenciar en el código de Linux ya sea a través de *commits* al código o bien a través de donaciones a la fundación que lo desarrolla.

licencias. O bien, para empresas grandes puede resultar aún más bajo, si cuentan con recursos para mantener un departamento de tecnologías de la información que les solvente dicho problema (véase: Capítulo II).

- Como estrategia mercantil. A través de apoyar el desarrollo de software libre, es posible disminuir el predominio de otra compañía en el mercado, sin caer en el riesgo financiero que implica desarrollar y comercializar un software bajo el esquema privado (para un trato más amplio de este punto, véase: Capítulo I).
- Añadir compatibilidad. Una razón para grandes compañías desarrolladoras de software para involucrarse en el software libre, puede ser liberar programas que provean compatibilidad de sistemas operativos con hardware (*drivers*[63]), así como piezas de software que permiten a distintas tecnologías de software interactuar unas con otras (*plugins*[64]), y que aseguren su predominio también en las plataformas libres sin necesidad de liberar código.
- Seguridad. La transparencia del código, permite que una empresa pueda saber cómo se está protegiendo su información (a través de algoritmos matemáticos específicos) y en dado caso de que surja algún problema tiene el respaldo de comunidades formadas por "miles de ojos" (Zúñiga & Camacho, 2004; Raymond, 1998), que permiten identificar agujeros en él y más importante aún, repararlos en el acto de ser descubiertos. En pocas palabras, no se está sujeto a que una empresa reciba el informe de alerta de seguridad, lo evalúe, le asigne prioridad, lo asigne a uno o varios técnicos y éstos se encuentren en condiciones de repararlo dependiendo del nivel de prioridad asignado, y donde su forma de repararlo no está sujeta a la crítica de pares no relacionados a nivel

63 En el caso del hardware, por ejemplo, el fabricante del chip acelerador de gráficos Nvidia o el fabricante de chips para tecnología Wifi, Broadcom, que han lanzado controladores para sus productos compatibles con Linux sin liberarlos bajo una licencia concordante con el software libre.

64 Como también es el caso de la tecnología para video en línea *flash,* utilizada en sitios populares como Youtube. Allende, Adobe (la empresa propietaria de los derechos de la tecnología *flash*) ha desarrollado *plugins* para navegadores que corren en Linux, sin liberar su código. Lo anterior, le permite mantener su primacía en todas las plataformas de sistemas operativos como el estándar de *facto*.

institucional, dado que se hace desde un código cerrado. Esto lo hace menos propenso al *malware*[65] y ataques exitosos de usuarios mal intencionados.

Al considerar las motivaciones que pueden llevar a las empresas a adoptar el software libre como una estrategia para afrontar las adversidades del mercado, es posible llegar a preguntarse no sólo por las implicaciones que tienen los aspectos sociales para el desarrollo particular o los motivos particulares que llevan a la práctica del software, sino las implicaciones de la práctica que conllevan dichas motivaciones para el panorama amplio que compone la industria del software. Por ejemplo, encontrar las implicaciones de utilizar el software libre como una estrategia mercantil para atacar el dominio de software propietario ya establecido en un nicho de mercado. De esta manera se puede colegir, la manera en que cambia esto la textura de las reglas del mercado y la cualidad de sus relaciones, así como la calidad de los productos.

c. La práctica del software libre

A continuación abordaremos de manera breve las características de la práctica en el desarrollo del software libre que lo diferencian de la forma propietaria de hacer software. Las condiciones de estas prácticas son posibles, dado el carácter intangible del software y a que la dinámica de la práctica desafían el modelo convencional de pensar el desarrollo (Deek & McHugh, 2007).

La primer práctica característica del desarrollo de software libre comienza en el acto de liberar código fuente al hacerlo accesible a quien lo demande.[66] Esto abre el acceso para que sus pares, personas con altos conocimientos

[65] *Malware*: es un término que refiere a todo tipo de software producido con la intención de realizar tareas no autorizadas y potencialmente perjudiciales para los intereses de aquel que lo ha puesto en funcionamiento (aunque muchas veces sin saberlo). Este tipo de software incluye entre sus tipos más conocidos: virus de computadora, gusanos, troyanos, *spyware* y *adware.* Y aunque sistemas operativos como Linux no son inmunes a los virus, nunca se han encontrado que afecten a estos sistemas en la misma proporción que afectan a los sistemas de Microsoft (Granneman, 2003). En este tema, el riesgo se encuentra reducido por la utilización de repositorios con validación de la integridad de los datos (*checksum* o *hash value*) que permiten confiar en que el software que se descarga no ha sido alterado*,* así como la estructura de multiusuario que se ha heredado de UNIX, donde un virus que infecte la totalidad del sistema requiere que sea ejecutado por el usuario raíz (*root*), el cual se encuentra protegido por contraseña.

[66] Hoy en día esto se realiza a través de hacer el código accesible para descarga en Internet.

especializados y quienes no guardan relación de ningún tipo con quien lo ha liberado y que por lo tanto, lo pueden evaluar sin compromisos (Zúñiga & Camacho, 2004).

No obstante, no cualquier código será acogido por la comunidad y comenzará a recibir sus contribuciones, éste debe tener dos características principales. La primera es que el código, al ser revisado por los pares, sea evidente ante éstos que es el producto es un aporte importante. Es decir, que sea difícil de recrear dado el "tiempo y el espacio cerebral" (Weber, 2000, pág. 23) de una persona creativa (lo cual no abunda) y que, por lo tanto, es valioso y digno de atención.

Por otro lado, debe ser un código que promete convertirse en algo interesante, es decir, se trata de código que no está terminado[67] y que puede ser mejorado. Al hacerlo, éste será mucho mejor con respecto a lo que ahora existe.[68]

Esto se resume en la práctica de "libere temprano, libera seguido y escuche a sus clientes" (*Release early. Release often and listen to your costumers*) (Raymond, 1998) lo que implica que una diferencia importante del software propietario con respecto al libre. Donde mientras para el usuario de

[67] Tal como puede inferirse del caso de Linux, en el que se puso a disposición de la comunidad un bosquejo de lo que después de convertiría en el kernel adoptado por GNU. Como marca Raymond (2000) en el caso del desarrollo del software libre se permite que las versiones tempranas tengan *bugs*, pero se espera que el usuario pueda tener una impresión más o menos certera del riesgo que implica el usarlo y los desarrolladores de lo que implica relacionarse con el proyecto. Por lo que si un código se encuentra en una etapa donde ni siquiera esto puede cubrirse, el proyecto no perdurará.

[68] Esta característica, fue delineada como parte de la cultura del regalo (*gift culture*) por Raymond (2000), donde se observa como un aspecto que sólo puede existir allende el bien es intangible en la sociedad de la postescasez. Desde este punto de vista, nos señala Castells, " las personas sólo pueden permitirse dedicar sus vidas a la creación intelectual cuando tienen sus necesidades materiales básicas cubiertas, y sólo en ese caso se puede practicar la cultura del regalo. De hecho, esto contradice la experiencia real de los *hackers* en países pobres, tales como Rusia o los de Latinoamericana. Precisamente en las situaciones de extrema pobreza, cuando las personas creativas no tienen acceso a recursos económicos, es cuando tienden a inventar sus propias soluciones, y lo consiguen. Los caminos sociales de la innovación son muy diversos y no pueden reducirse exclusivamente a las condiciones de vida materiales. Pero lo que es común a la cultura *hacker*, en todos los contextos sociales, es su tendencia a reinventar modos de comunicarse con y mediante los ordenadores, construyendo un sistema simbiótico de personas y ordenadores que interactúan a través de Internet" (Castells, 2001, págs. 65-66). Donde la cultura *hacker* es, "en esencia, una cultura de convergencia entre los humanos y sus máquinas en un proceso de interacción sin trabas. Es una cultura de creatividad tecnológica basada en la libertad, la cooperación, la reciprocidad y la informalidad "(Castells, 2001, pág. 66).

software propietario una "versión 1.0 significa «No toques esto si eres prudente»[69]; en el mundo del código abierto, se entiende más como «Los desarrolladores se encuentran dispuestos a apostar su reputación en esto»"[70] (Raymond, 2000) por lo que muchas aplicaciones tienden a permanecer mucho más tiempo en versiones previas a la 1.0. Si bien, esta práctica puede ser utilizada en el ámbito del software propietario, el ritmo de las actualizaciones es mucho más corto en el software libre, en el software propietario también es posible escuchar a los usuarios a través de reportes de *bugs*[71] (Valloppillil, 1998).

Otro aspecto, es la práctica de la contribución al código original. Si se trata de contribuciones voluntarias, se requiere la práctica de su administración, es decir, la forma en que se mantienen interesada y activa a la comunidad que crece en torno al proyecto. En este punto resulta importante la forma en que se racionaliza la participación, se otorgan reconocimientos a través de motivaciones como alimentar el *ego,* al premiar con exhibición de la mejora diaria o constante del trabajo que han realizado los contribuyentes[72] (Raymond, 1998). Al liberar temprano, la necesidad de resolver bugs atrae a los usuarios experimentados, no los ahuyenta (Deek & McHugh, 2007), como sucede en el software propietario, donde los usuarios esperan que funcione sin problemas cualquier programa por el que han pagado una licencia.

La colaboración colectiva hace que en la práctica del software libre exista más posibilidad de que "con muchas miradas, todos los errores saltarán a la vista" ("*given a thousand eyes all* bugs *are shallow*") (Raymond, 1998), lo que

---

69 "(...)a mí me tocó la primera versión más o menos buena de Windows, las 3.1, que requerías 20 megas en disco y en memoria 64k y la Mac, con menos memoria corría más eficiente que Windows" (Entrevista 7 a miembro de AMESOL: Marzo 2010).

70 Como ejemplo de esto, podemos pensar en que Windows fue adoptado masivamente hasta su versión 3.0, para administrar MS DOS, como interfaz gráfica, dado que ofrecía más funcionalidades y mejor uso de memoria que versiones anteriores. Por otro lado, el *kernel* Linux se hizo disponible para descarga en su versión 0.01 desde Septiembre de 1991, con 10 239 líneas de código, y hasta Marzo de 1994, la versión 1.0.0 fue liberada con 176,250 lineas de código, con esto nos podemos dar una idea de todo el trabajo que fue necesario antes de que los desarrolladores pudieran apostar sus reputaciones en el *kernel* Linux.

71 Sin embargo, hay que hacer notar que aquí no se hace caso a los usuarios con menos experiencia, sino a los más experimentados.

72 El propio Raymond (2000) apunta que existe resistencia de los *hackers* a admitir el *ego* como una motivación de peso.

genera la posibilidad de que cuando se detecte algún problema en la programación, alguien que tenga interés en resolverlo entienda porqué sucede y pueda resolverlo, lo cual, tiende a pasar con gran frecuencia y de manera rápida en proyectos donde existe gran participación.

Él éxito de estas prácticas, las cuales estructuraran formas de organización y han demandado la creación de nuevos medios técnicos, se puede ver reflejado en la cantidad proyectos alojados en diversos repositorios o directorios de proyectos de software libre, tales como SourceForge[73], freshmeat, GNU, Apache Software Foundation, CPAN, etc. A mediados del año 2005, freshmeat tenía cerca de 40 000 proyectos y GNU más de 4 000, donde el más grande era, y aún lo es hasta febrero de 2009, Sourceforge.net con más de 230 000 proyectos registrados y con más de 2 millones de usuarios inscritos (Geeknet, 2010).

En el año 2010, al rededor del 49% de los proyectos alojados en SourceForge tienen algún tipo de licencia GPL y sólo alrededor de 5% tienen licencia BSD, y el resto son diversas licencias que pueden ser consideradas software libre. De esta forma, el 72% tienen una licencia que ha sido aprobada por la Open Source Initiative (OSI) como fuente abierta (*open source*).[74]

Al principio, como podemos observar, la práctica del software libre se basaba primordialmente en un modelo puramente colaborador de carácter voluntario, empero, hoy en día se realiza en la práctica profesional de desarrolladores empleados por corporaciones que buscan, en última instancia, la realización de ganancia. Tener presentes estas prácticas, nos serán útiles en el momento en que abordemos la práctica del software libre como estrategia empresarial dentro de las empresas mexicanas, ya que pasaremos a realizar el análisis de este fenómeno a través de los conceptos que hemos delineado en el primer capítulo, como veremos en el último capítulo de este tesis.

[73] Éste sitio es controlado por el Open Source Technology Group, la que a su vez es propiedad de VA Software que administran freshmeat (Deek & McHugh, 2007).

[74] El cálculo se hizo conforme a los datos de Geeknet (2010), tomando la adición de los programas registrados bajo la totalidad de las categorías, incluyendo los que tienen la categoría “sin categoría” para un producto de 225678, menos de los que proclamaba tener el sitio un año antes. Éste total se comparó con los números que ofrece para cada tipo de licencia, así se agregaron la GPL y la GPLv3. La OSI (2010) aprueba la GPL como una licencia de fuente abierta, y por lo tanto se incluye en su porcentaje.

Conclusiones

Ubicar de manera histórica, el cómo surgió y se potencializó la forma de producción del software libre, nos permite ubicar que su emerger ha estado imbricado por un proceso dialéctico, en el que se vislumbra por un lado, que su caracterización como conocimiento un bien que no es escaso, por el otro, se distingue un proceso de restricción legítima del conocimiento vía instrumentos económico-legales.

La primer forma de producción original del software sería puesta a un lado por el dominio del segundo, que constituiría propiamente la industria del software, me refiero al sometimiento de la que compartía el código fuente al modelo de licenciamiento. Ambas formas de apropiación del conocimiento, sugieren modelos distintos de realización de ganancia, es decir, de subsunción en el modo de producción capitalista, el primero como trabajo que realiza un servicio, el segundo como un producto. Uno como conocimiento codificado, el otro solamente como conocimiento objetivado (tal como vimos en el primer capítulo).

Ahora bien, también se puso en claro que la organización productiva del software libre, fue anterior al auge de las tecnologías de la información como Internet, como fue el caso de la BSD. Que fue producto de continuar con la práctica básica de compartir el conocimiento, tal como lo hacían los primeros programadores de software y que utilizando dicha práctica se montó una organización en torno a un proyecto de software que producía de manera sostenida. Esto es, con ciclos definidos lo que hoy conocemos como software libre.

Esta organización de desarrollo es jerárquica, donde existen aún “los grandes magos encerrados a piedra y lodo”, quienes deciden que código es lo suficientemente bueno como para entrar a formar parte del software que se distribuirá “oficialmente” en la siguiente distribución.

Pero encierran una gran diferencia con otros modos de desarrollo, no se encuentran supeditados a formar parte de una sola institución o empresa. Y por lo tanto, las revisiones del software son echas por colegas y las jerarquías se

encuentran así, regidas por meritocracia, no por aquel que dispone de una posición burocrática superior (como sería el caso del que se desarrolla en una empresa).

Internet es un catalizador que permitió la explosión en el desarrollo de software libre. Que tuvo que echar mano de medios técnicos, para potencializar la organización de desarrollo del software libre complejo, como fue el software controlador de versiones, como fue el CVS, Subversion, Bitkeeper o *git* y el diseño modular. El *kernel* Linux es uno de los ejemplos más prominentes, cuyo crecimiento obligó a su fundador, delegar su posición como líder, su posición de "dictador benevolente".

El proyecto Linux ha sido exitoso, porque es masivo, se trata de un proyecto jerárquico con una base muy amplia y que requirió avances técnicos. Del mismo modo que el taylorismo fue potencializado por el fordismo, la estructura de producción del software libre fue potenciada por internet en general y particularmente por los *version control systems y* el diseño modular.

Por un lado, el diseño modular fue fundamental al permitir dos aspectos básicos, el primero es la posibilidad de añadir funcionalidades al kernel sin la necesidad de reescribirlo, y la segunda es que los programadores podían hacerlo sin interferirse mutuamente. Por otro lado, los *version control systems*, hicieron posible, coordinar las contribuciones de manera más ágil, cuyo desarrollo convergió con las condiciones que permitieron acelerar el desarrollo del *kernel* Linux.

Al principio, con un esquema técnicamente centralizados de desarrollo, en programas como el CVS. Empero, posteriores desarrollos en programas como Subversion, disminuyeron la verticalidad un peldaño al permitir experimentar con el código que se albergaba en el servidor principal.

Posteriormente, el desarrollo de BitKeeper, GNU Arck y *git*, permitieron a los programadores trabajar asincrónica e independientemente, lo que habilitó eliminar la necesidad de un repositorio central. Donde se puede apreciar que los avances técnicos han permitido que el desarrollo del software libre se descentralice y que la base de sus pirámides jerárquicas se ensanchen.

Estas jerarquías en un proyecto grande, como BSD o Linux, se componen de un *core team, commiters, developers* y los usuarios, todos ellos conforman la comunidad, del software libre. Con papeles definidos de acuerdo a sus prácticas en la comunidad, a sus contribuciones y a sus logros.

Así, las comunidades son geográficamente descentralizada, por ejemplo, Linux, iniciado por un finlandés, no puede proclamar que se trate de un producto de aquel país, son proyectos que no tienen anclaje territorial o afiliación nacional. Como conocimiento, se trata del mismo software el que se descarga en japón que el que se descarga en cualquier otro lugar del mundo donde exista una conexión de Internet. Tal vez tarde unos cuantos segundos en Japón dado su ancho de banda, de lo que puede tardar en la ciudad de México (un par de horas), pero al final, el software que hace funcionar el hardware es el mismo. Y se puede aprender de él, se le puede modificar y se le puede redistribuir exactamente igual en ambos lugares.

Apropiarse del código tiene dos dimensiones: la primera es como conocimiento objetivado, como una cosa. Se puede pensar como un conocimiento que aparece alienado de manera legítima, cuyo costo es ínfimo. Por otro lado, tiene otra dimensión, que es su apropiación como un “bien club”, que requiere su apropiación como conocimiento codificado. El costo de esta dimensión es más alto que el primero, porque requiere inversión en tiempo y simplemente no todos los integrantes de las sociedades quieren y/o pueden ser programadores.

Ahora bien, se han presentado dos formas en que pueden pensarse las motivaciones para adoptar el desarrollo del software libre. Por un lado, las motivaciones individuales de los programadores para hacer software libre tienen diversos aspectos, que pueden ser económicos, técnicos, políticos, psicológicos o sociales. Éstas, han sido analizadas desde una perspectiva de costo beneficio, especialmente desde la teoría del *rational choice*. Del mismo modo, se han presentado posibles motivaciones de las empresas para acoger su desarrollo.

Desde una perspectiva marxista, crear valor de uso es motivo suficiente

para crear software libre, un bien que no se desgasta con el tiempo, que seguirá funcionando y que si se otro miembro le hace mejoras, estas podrían beneficiar a quien lo creo en primer lugar. De la misma manera, mejorar código permite no empezar de cero.

Sin embargo, no se debe perder de vista, que además de las motivaciones anteriores, también una motivación en donde se puede encontrar imbricada la historia del software libre, me refiero a la motivación en la que significa "oponerse a formas egoístas de hacer software". La que parece ser la más trivial, encierra la lógica misma que inició el software y que se opone más tajantemente a la forma en que se consolido la industria del software como un negocio altamente lucrativo.

Por otro lado, las motivaciones de las empresas, también dejan entrever aspectos interesantes. Entre las que se puede señalar que faculta la descentralización del riesgo de invertir, al permitirles estandarizar el mercado mientras se encuentran seguras de que su inversión no será controlada por otra compañía.

De esta manera, podemos distinguir cinco intereses: a) El costo por licencias es cero y el pago de mantenimiento puede resultar menor. También, b puede funcionar como una estrategia mercantil, al descentralizar el costo de desarrollar y competir con una empresa rival, donde; c) se crea una plataforma paralela (Linux) que es capaz de aprovechar el desarrollo y *expertise* que se había consolidado para UNIX durante décadas precedentes, con la seguridad de que no serán apropiados por ninguna empresa y que, eventualmente les permite hacer las mejoras que se requieran sin depender ninguna otra empresa. d) Continuar su monopolio, en el mundo del software libre, y; c) la seguridad que se deriva de miles de ojos que pueden observar el código y permiten que al presentarse un problema, éste sea reparado en el acto o bien, se de la voz de alarma que permita que aquel sea reparado con celeridad.

Lograr establecer una comunidad, tan exitosa como la que se ha configurado en torno al *kernel* Linux, requiere de ciertos rasgos, como lo es lograr la consolidación de una comunidad de personas capaces. Estas

comunidades se forman en torno a piezas de código que resultan prometedoras, donde es posible apreciar que se trata de un aporte que vale la pena considerar, ya que resulta evidente que es un código que ha requerido tiempo y espacio cerebral de una persona creativa.

Dichas aportaciones, se hacen dado que no se encuentra terminado, pero que su resultado promete convertirse en algo interesante. Este hecho, atrae a los *hackers*, no los ahuyenta y promueve su participación. Y entre más participación hay, se resuelven problemas de programación con más frecuencia *(bugs).* Por consiguiente, liberar temprano y vigorosamente, ha probado tener sentido en el mundo del software libre, donde los proyectos han llegado a sumar más de 100 000 y contar con millones de colaboradores.

En el siguiente capítulo, veremos como se usa el software libre como estrategia para producir ganancias. Dichas estrategias, comprenden la reincorporación de modelos de servicios proporcionados al rededor del software así como la introducción de modelos híbridos.

En un ambiente, donde el software crece en complejidad, se requieren servicios de soporte técnico, que comprende mantenimiento, capacitación y ofrecer soluciones. Lo que se puede lograr, a partir de código ya desarrollado y maduro, que se orienta más a consultoría y puesta a punto.

Lo anterior, lleva a disipar la intensidad de la producción del conocimiento de las empresas que usan software libre y les orienta más a su adaptación. Por otro lado, también emergen los modelos híbridos, que generan herramientas propietarias que permiten realizar tareas con el software libre más fácilmente, pero que no se encuentran liberadas bajo una licencia *open source*.

Capítulo III: Valorización del conocimiento y realización de ganancia en dos modelos producción del software

En el capítulo I, se estableció que desde una visión histórica amplia es posible apreciar que el conocimiento ha adquirido paulatinamente preponderancia en la valorización de la producción y generación de ganancia. Diversas transformaciones, en la división social del trabajo e innovaciones en tecnologías, permiten comunicarse de manera instantánea y procesar en segundos la información que se comunica. Esta situación ha permitido que el conocimiento sea pieza fundamental de la sociedad capitalista globalizada en la que vivimos actualmente.

Conceptualizar rasgos que componen la realidad ayudan a identificar como se han dado dichas reconfiguraciones, como lo son: a) La forma en que se produce el conocimiento útil en el proceso productivo, b) las distintas formas en que se realiza la apropiación del conocimiento y c) la forma en que se realiza la creación de valor.

En el mismo sentido, se ha establecido que el devenir histórico ha hecho posible que aparezcan nuevos tipos de bienes. Éstos, se han tratado comercialmente como mercancías materiales o tangibles, pero que más bien devienen de la expresión formal de conocimiento tácito. Brindándoles así, propiedades que lo acercan más a la concepción de un bien inmaterial o intangible.

Un ejemplo de lo anterior, es la empresa de software privado, ya que ésta lo cambia en el mercado como si se tratase de un bien tangible y en cambio, se le protege como un bien intangible. Donde se apuesta por la recuperación del costo de desarrollo, producción y distribución en la venta de licencias de uso.

Ante el modelo que se acaba de exponer, ha aparecido un segundo modelo de empresas que han apostado por estrategias en el mercado que incluyen el software libre. Donde el bien intangible del software no se encuentra

sujeto al mismo tipo de protección económico legal. Y reconocen, que la forma de producción del software libre requiere de la salvaguarda de condiciones que alienten ser compartido y modificado.

En el capítulo II observamos en detalle cómo fue que se organizó el proceso de producción del software libre, en un contexto histórico con base en ejemplificaciones concretas. Asimismo, se esbozaron las diversas motivaciones que han orientado varios estudios desde el ámbito económico y las prácticas sociales que hacen posible su existencia.

A través de este capítulo, nos acercamos a la concepción teórica de los modelos del software privado y libre, a la vez que se presentan casos de empresas que practican ambos modelos. De este modo, en el aspecto teórico nos concentramos en la forma en que se realiza el desarrollo del software y la forma en que se realizan ganancias en cada modelo. Y al final en los casos concretos representativos del modelo de la empresa privada del software se presentan los aspectos que presenta Microsoft y en el segundo, a empresas que han invertido en el software libre, como IBM, Red Hat y Novell.

Así, en el modelo de la empresa privada, tenemos que la producción de conocimiento se realiza de manera jerárquica.[75] Al tiempo que responde de manera pasiva a la retroalimentación del mercado y que se orienta a crear necesidades (Revelli, 2004).

Donde además, existe apropiación de conocimiento que se sustenta en el registro de patentes[76] y derechos de autor. Es decir, el modelo de valorización se basa en la apropiación de la objetivación del conocimiento. O dicho de otra manera, en la apropiación de la expresión formal del conocimiento tácito de sus trabajadores, en este caso programadores (véase: Capítulo I).

Y por último, se realiza ganancia a través de la venta de licencias para el uso del software, la cual asemeja a la venta de cualquier mercancía tangible. Con la diferencia de que prohíbe hacerle modificaciones o redistribuirlo. Tal

[75] En este modelo, se advierte de manera general muchas semejanzas con la forma de producción taylorista-fordista, que apela a tiempos y formas de organización jerárquica tal como veremos más adelante.

[76] Dependiendo del país en el que se encuentra, se pueden aplicar patentes a la protección del *software (Stobbs, 2000)*, sin embargo, como se ha notado en el primer capítulo es más común el uso de derechos de autor o también llamado *copyright* en el derecho anglosajón.

como se podría hacer con las mercancías de aquel primer tipo (Lessig, 2004).

En segundo lugar, tenemos el modelo de empresa que ha adoptado el software libre. Éste tiene como principal característica aprovechar el conocimiento que se ha sintetizado en software de manera colaborativa. Principalmente a través de Internet, donde el propio procedimiento que ha hecho posible la existencia del código desalienta que el código sea cerrado. E incluso, se han diseñado estrategias de licenciamiento que previenen que esto pase[77].

Donde la valorización del software se descentraliza de la empresa, si bien puede seguir existiendo desarrollo de software que es controlado y orientado por ésta. De esta manera, se apuesta por abrir el proceso de desarrollo a comunidades de programadores que encuentran motivos suficientes para formalizar y compartir su conocimiento al codificarlo en software (véase: Capítulo II).

En este modelo, la ganancia se realiza de manera más sutil, no a través de la venta de licencias de uso de programas, sino de la venta de servicios, como son consultorías, soporte, adaptación, puesta a punto, capacitación. Todas ellas, como veremos aquí, tienen sentido en un contexto que les brinda sentido.

De esta forma, la estructura del capítulo se orienta a contestar las siguientes preguntas en ambos modelos, en primer lugar ¿cuáles son las implicaciones del conocimiento para valorizar lo producido en cada uno de estos modelos? Y en segundo lugar ¿cuáles son las estrategias que se utilizan en la sociedad del conocimiento para lograr realizar ganancia?

Para ello, se realizará un retrato de su propia dinámica. O sea, un análisis de como se reconfigura en cada uno de los modelos las implicaciones de diversas formas de creación de valor. Con esto, será posible apreciar el cómo se realiza la práctica social de valorización y de la apropiación privada de

[77] Tal es el caso de la GPL que evita que el código sea cerrado si alguien le ha realizado modificaciones, se trata de una licencia que obliga a compartir las mejoras que se hagan al *software*, a diferencia de la licencia BSD, que permite que el *software* sea apropiado, se cierre el código y este sea redistribuido de manera privada se le hayan hecho o no modificaciones.

la ganancia.

1. Modelo de valorización en la empresa de software privado

El modelo de la empresa de software desarrollo de manera privada tiene las características de producción, distribución y consumo de cualquier otra mercancía. A pesar de ello, no se trata de cualquier mercancía, ya que como hemos señalado, se trata de conocimiento que se ha sintetizado en software. Y el cual, es el resultado de la formalización de habilidades y experiencias por parte de los trabajadores.

El conocimiento sintetizado, tiene la particularidad de ser "lógico no físico, no se fabrica, se desarrolla y no se degrada con el tiempo" (Mochi, 2006, pág. 202). Éste, también es susceptible de ser apropiado a través de regímenes legales, en el caso del software, predominantemente a través de los derechos de autor y que, a la vez, legitiman un modelo de comercialización, como lo es la venta de licencias de uso.

Actualmente, el licenciamiento del software es el modelo dominante de realización de ganancia en esta industria. Se basa en que el costo de su desarrollo debe ser recuperado a través de las ventas de unidades individuales, que son distribuidas a los consumidores. Por ello, normalmente se asume que es imperativo para quienes lo licencian[78], que sea protegido por marcos legales, para que no sea reproducido libremente por cualquiera y de esta manera dejar de percibir renta (Dalle, David, Ghosh, & Steinmueller, 2004).[79]

Los marcos legales que hacen posible esta estructura de realización de ganancia son eficaces, en tanto permiten restringir legítimamente el conocimiento. Como también, al sostener estrategias sociales que posibilitan valorizar conocimiento, apropiarlo y conservar exclusividad de control sobre el mismo, es decir de su monopolio.

En la primera sección de este apartado, abordaremos la forma en que se crea la valorización del software, es decir, la transformación del conocimiento de

[78] Para observar experiencias concretas, que contradicen esta suposición véase: Capítulo IV.

[79] Dicho modelo, se asemeja al que había seguido la industria discográfica durante al menos 60 años antes de que madurara la industria del *software* e incluso, 500 años antes si se considera la industria del libro (Thompson, 1995).

tácito a su formalización. Esto es, el paso del conocimiento de los tecnólogos, al que se encuentra objetivado en medios electrónicos. Donde abordamos el régimen de organización para su desarrollo, que pueden ser institucionalizados y jerárquicos, ahí veremos su relevancia para la valorización y la ganancia. Y por último, observamos la mecánica que guarda su valorización como apropiación de la ganancia.

En la segunda sección, apreciamos como es que existen estrategias de realización de ganancia en mercados que existen realidades concretas. Que son posibles por la existencia del monopolio del conocimiento codificado. Lo cual, le permite ejercer poder de mercado.

Por último abordaremos un caso concreto encarnado por la empresa Microsoft. Allí, señalaremos la manera en que realiza la extracción del conocimiento de sus empleados. Como también, la forma en que ha practicado estrategias que llevan a perpetuar su modelo de realización de ganancias.

a. Creación estandarizada y jerárquica del *software*

Los principios económico-legales, que permiten la producción y valorización del conocimiento a través de normarlo y estandarizarlo, permitieron consolidar históricamente la industria del software. Que ha pasado de ser artesanal a convertirse en una serie de procesos uniformes y jerárquicos. Dichos procesos, conllevan división social del trabajo en tareas específicas que asemejan el modelo de producción establecido en la gran industria por el paradigma taylorista-fordista. A continuación, abordamos un ejemplo de como se ha homogeneizado el desarrollo de software y la forma en que existe organización del trabajo en estos modelos.

i. Estandarización: la institucionalización del desarrollo del software

Los procesos de estandarización se encuentran dirigidos a institucionalizar la producción del software. Tal es el caso del "Modelo de Madurez de Capacidad del software" (SEI; Software Engineering Institute) que tienen por objetivo lograr la mejora continua del mismo (Bodas, 2003) (Tabla 7).

Tabla 7: Niveles de institucionalización del proceso de desarrollo del software

Nivel 1	Inicial	El proceso de software es impredecible y poco controlado. Esto no significa que una organización no produzca buen software, sino que el coste (financiero, humano, temporal, etc.) es demasiado alto tanto para los productores como para los usuarios.
Nivel 2	Repetible	Disciplina básica en la gestión de procesos basada en la repetición de tareas aprendidas previamente. Ya hay una planificación en términos de coste, calendario y requisitos.
Nivel 3	Definido	El proceso es estándar y consistente, se conoce lo que hace que el proceso de software tenga éxito y se aplica a toda la organización.
Nivel 4	Gestionado	El proceso del nivel 3 es medido y controlado cuantitativamente, está implementado en toda la organización.
Nivel 5	Optimizado	Existe una evolución continua en la optimización del proceso.

Fuente: Elaborada a partir de Bodas (2003).

Como vemos, en este proceso existe contenida su propia negación, puesto que se diseña para cambiar continuamente. Tal como se observa, en el "Nivel 5", se busca la continua optimización del proceso. Se trata de ajustar el proceso caótico de creación a la disciplina industrial de los plazos.

Para evitar disciplinas rígidas, se alienta la consideración de procesos dinámicos que inciten la creatividad, "el reto consiste en administrar actividades que nunca han sido ejecutadas en el pasado y tal vez, nunca sean repetidas en el futuro" (Mochi, 2006, pág. 203). Pese a lo anterior, el proceso de desarrollo del software ha requerido la construcción de prácticas para constituir el conocimiento codificado, y éstas, se han inspirado en modelos jerárquicos, similares a las tayloristas-fordistas o que incluso pueden ser consideradas como postfordistas, que como veremos, también presentan inconvenientes.

ii. La jerarquía al interior del desarrollo del software

La estructura de "Programador en Jefe", es un ejemplo de como se adoptó una estrategia jerárquica en la producción de software. Este modelo fue puesto en práctica por IBM a finales de 1960 donde trabajó Harlan Mills (1919-1996), quien propuso un equipo, compuesto de:

1. Un "programador en jefe" quien se encuentra encargado del desarrollo principal.
2. Un "programador de apoyo" que critica el trabajo del jefe de programación,

asistente de investigación, tiene contacto técnico con grupos externos y respalda al jefe.

3. El "administrador" quien maneja cuestiones administrativas, tal como dinero, gente, espacio y máquinas. Siempre supeditado a la palabra final del jefe pero también lo libra de lidiar con aquellas diariamente.
4. El "*toolsmith*", quien es responsable de crear herramientas encargadas por el Jefe. Esto significaría encargarse de mantener el ambiente de construcción y crear *scripts,* etc.
5. El equipo se encuentra apoyado ocasionalmente por un "abogado del lenguaje" que apoya al jefe al contestar preguntas sobre el lenguaje de programación que se encuentra utilizando (McConnell, 2008 T. del A.).

Hoy en día varias de funciones descritas anteriormente son desarrolladas por personas que no programan. Tal es el caso de especialistas en documentación, probadores de programas (*testers*) y gerentes de programación. También, el Internet ha reducido la importancia del "abogado del lenguaje", cuestión que se solventa con una búsqueda en la red (McConnell, 2008).

Esta estructura organizacional de desarrollo, puede considerarse heredera de las que había propuesto la taylorista-fordista, puesto que su jerarquía es vertical. Pero también, es eficiente para desarrollos de software relativamente pequeños, no obstante, también suele tener inconvenientes. Ya que se ha observado, que cuando los proyectos de software fracasan "(...) se debe a problemas en el trabajo en equipo y no a aspectos técnicos" (Mochi, 2006, pág. 205).

De esta forma, los problemas que se pueden presentar, se resumen en la llamada "ley de Brooks". Básicamente esta ley, dice que: Tan pronto se alza el número de programadores en un proyecto, el trabajo realizado se incrementa linealmente. Pero, la complejidad de la comunicación y la propensión a los *bugs* se incrementa de manera geométrica, es decir, con un factor cuadrado. Esto se atribuye a que aumenta la comunicación potencial entre desarrolladores que escriben código de manera individual (Brooks, 1995; Weber, 2000).

En la práctica, este problema se presenta cuando los directivos de la empresa desarrolladora de software se dan cuenta de que existirá retraso. Debido a esto,

en la mayoría de los casos proceden a añadir personal a la configuración del programador en jefe. Debido a esto, se eleva la complejidad de la comunicación humana entre los participantes del proyecto. De este modo, con cada persona que se aumenta, el tiempo necesario para terminar la tarea se incrementa exponencialmente.

Por otro lado, en sistemas de gran complejidad -como lo son sistemas operativos- es necesario que los programadores se acoplen en equipos de trabajo mucho más grandes. Este incremento en la complejidad, se resuelve a través del diseño del propio sistema. Al igual que vimos en el capítulo II, estos sistemas se encuentran construidos a través de módulos lógicos que permiten a los programadores dedicarse a sólo crear y mantener ciertas piezas de código, sin alterar la estructura total del sistema (Flynn & Mchoes, 2001). Esta forma de organizar el desarrollo se asemeja al modo taylorista de producción, al parcelar el conocimiento en módulos distintos al igual que Taylor lo hacía sólo para las tareas asignadas.

Como veremos más adelante, el software que se produce de manera privativa, no necesariamente circula al interior de las grandes empresas, por lo que ni siquiera los módulos o piezas de código pueden ser reutilizados en diferentes desarrollos al interior de la propia empresas. Tal como se puede colegir de la lectura de Vinod Valloppillil (1998) (véase: El siguiente subapartado).

Ahora bien, la transformación del conocimiento tácito en formalizado, puede concretarse a través del uso de varias metodologías de desarrollo que pueden utilizar varias aproximaciones. Éstas pueden servir para guiar su desarrollo y las actividades necesarias para crearlo, tal es el caso del modelo de cascada, espiral y *agile.* Este último, nace como una reacción en contra de los modelos de desarrollo inflexibles, que no apuestan por revisar las anteriores fases de desarrollo (Beck & Beedle, 2001)

A pesar de esto, todas estas formas de producción son centralizadas si se considera que a fin de cuentas existe un plan de ruta que orienta su desarrollo, que traza lo que el software debe hacer. Y dado que, se desarrollan

en ambientes interpersonales, se presenta el problema de la complejidad comunicativa entre sus programadores, lo que produce retrasos y propensión a fallas en el código.

Al momento de realizarse la transformación del conocimiento en software, esta apropiación es jerárquica. Ya que su apropiación (la propiedad) del mismo, tiene mayor probabilidad de ser detentada por aquel que tiene los derechos de autor, no en quién objetivó su conocimiento en el software.

Así, pensado el proceso de valorización del software, como proceso de apropiación, en el software privativo, siempre es jerárquica. Por esta razón, puede decirse en palabras de Marx (2002), que el programador "(...) no se apropia del producto de su propio trabajo, que ese producto se le presenta como propiedad ajena; a la inversa, que el trabajo ajeno se presenta como propiedad del capital" (pág. 431).

Más allá de las metodologías de transformación del conocimiento tácito en explícito y sus inconvenientes[80], siempre que se realiza en su forma privativa, su valorización se encuentra dada por los regímenes legales que impiden su difusión como conocimiento. Lo que a su vez, legitiman su apropiación como propiedad del capital. Aquí, su ruta de desarrollo siempre se encuentra dada por el interés del capital, es decir, la apropiación siempre es jerárquica y centralizada.

iii. Abstracción de los pasos para valorizar software y su modelo de realización de ganancia

Las empresas privadas del software aspiran a administrar el proceso creativo de codificar conocimiento y formalizarlo en software. Para lograr esta administración se ha aspirado a la institucionalización del proceso organizado en estructuras jerárquicas de desarrollo, ya sea en torno a la estructura del trabajo o a la estructura técnica del diseño del software. Las cuales, tienen como eje rector la restricción del conocimiento, por un lado los grupos de programación al dividir el conocimiento sobre el software en diversas tareas y el

[80] De carácter interpersonales, dados por la complejidad de la comunicación entre ellos, como lo propone la "ley de Brooks".

segundo al parcelar el conocimiento del software sólo a los módulos asignados.

Ahora bien, las diversas metodologías jerárquicas que existen en la empresa privada del software y la forma en que realizan ganancia, siguiendo a Dixon (2009) en posible abstraer en una serie de fases, sin importar la metodología que desarrollo que se utilice:

1. La planeación;
2. Fase de implementación: en la cual se realizan pruebas y se documenta;
3. Comercialización: se dedica a incorporarse a la venta en el mercado o en su caso a la aplicación en un ambiente de trabajo;
4. Capacitación a los usuarios;
5. Seguimiento y corrección de problemas a través de un sistema de detección de *bugs.*

Como se puede apreciar, estas fases podrían seguirse desde el modelo de desarrollo del “Programador en Jefe”, pero no importa que no se siga la misma estructura de trabajo, las tareas que se han presentado deben ser cumplidas de cualquier forma, con lo cual, se afianza la institucionalización del proceso de software comercializado a partid de licencias. Lo anterior, también puede permitir que se aspire a que el producto del software cumpla con estándares de “calidad”, aunque el resultado no siempre cubra las expectativas de calidad en su consumo, *verbi gratia:* la “ley de Brooks”.

De la misma forma , la producción en la empresa privada del software requiere en general recursos humanos y capital que se elevan conforme se eleva la complejidad del software que desarrollan. Pero también, se puede abstraer toda producción de software privado al modelo al modelo que propone Dixon (2009):

1. Ingenieros que tienen dos roles: el primero como creadores de software, y el segundo como participantes en el programa comercialización.
2. El gerente del producto “es propietario” del producto y del diseño que ha de seguir, así como tiene la responsabilidad de crear y ordenar el requerimiento de ventas, comercialización y clientelas. También actúan como un intermediario entre la ingeniería y los grupos encargados de la

comercialización. Esto se hace por dos razones: conservar a los ingenieros concentrados en escribir software, y el control del flujo de información de los clientes a los ingenieros. Los gerentes del producto también describen como es que las características son convertidas en un "producto completo".

3. Las ventas, comercialización, soporte y departamentos de servicios se han centrado en entregar todo un producto a los clientes.
4. Es ese programa de comercialización lo que crea todo el producto que la mayoría de los clientes requiere. Los ingenieros no van al mercado, ellos son participantes (usualmente reticentes).
5. El cliente se encuentra indirectamente involucrado en el proceso de crear el software.

Este modelo de producción, presenta como se pasa de la sintetización del conocimiento a un producto completo. Lo cual es posible porque se basa en la apropiación legitimada por un régimen económico-legal, que sustenta su realización de ganancia a través de su práctica. Sin embargo, ser un régimen legítimo no es suficiente para ser practicado en la realidad. La complejidad de la realidad demanda que este modelo sea complementado con estrategias que implican ejercicio de poder, legitimado si se quiere por el régimen legal, pero que sin lugar a dudas orientan las prácticas de sujetos que son parte del mercado, las cuales atendemos en el siguiente apartado.

b. Estrategias de realización de ganancia

El modelo de desarrollo del software, permite ir al mercado con un producto terminado, el cual tiene la característica de ser conocimiento codificado que se encuentra formalizado. Esto le permite ser apropiado al ser estudiado, modificado y redistribuido, brindándole monopolio sobre ese conocimiento[81],

[81] Si bien, se puede argüir que en cualquier mercado existe monopolio de bienes intangibles, como fórmulas farmacéuticas o diseños industriales protegidos por patentes, y que garantizan un periodo de exclusividad para realizar ventas que cubran las inversiones que se han hecho en investigación y desarrollo, y además conseguir ganancia, la diferencia esencial es que el software, al ser un bien intangible y ser al mismo tiempo el bien que se transa, es susceptible de ser reproducido sin esfuerzo y que su consumo no implica la destrucción del software original.

que producido socialmente es apropiado de manera privada.

Si bien, los modelos de desarrollo de software permiten abaratar costos de desarrollo, al reducir tiempos y mejorar su “calidad”, también existen prácticas que se basan en la estructura característica de un mercado concreto. Y son éstas las que le permiten mantener una demanda constante de software privado y por lo tanto, la demanda de su desarrollo.

En este apartado exploramos tres estrategias que permiten realización de ganancia a partir de aquel tipo de software en el mercado. La primera es la que se deriva de la existencia del monopolio de *facto* del conocimiento, legitimado por el régimen legal imperante. Llámese derechos de autor, *copyright*, patentes o propiedad intelectual, puesto que todos estos regímenes hacen posibles condiciones que permiten realizar diversas estrategias de comercialización del software.

En segundo lugar, y como derivación de estos monopolios de *facto* del conocimiento, encontramos lo que se conoce como el *lock-in* tecnológico. Que si bien no es exclusivo del software, adquiere connotaciones específicas cuando se presenta en la comercialización, distribución y consumo, como si se tratase de cualquier producto tangible.

Por último, veremos en qué consiste la estrategia conocida como "adoptar, extender, y extinguir" (EEE[82]). La cual, fue introducida por Microsoft, pero empleada también por otras empresas. Y que, a diferencia de la anterior, únicamente se puede aplicar en esta industria.

Este abordaje, nos permitirá, en una instancia posterior, abordar el porqué es relevante la forma en que se valoriza el conocimiento en la realización de la ganancia. Ya que el software libre, se construye en un espacio social dado, que presenta un mercado en el que se distinguen las características que describimos a continuación.

[82] Del Inglés *Embrace, Extend and Extinguish*

i. Reproducción de los monopolios del software

Una de las características primordiales del capitalismo es la búsqueda de ganancia, para ser reinvertida en el propio capital. Después de esto, si retomamos la idea que apunta Schumpeter (1983), el capitalismo es un proceso evolutivo y orgánico, que sostiene su reproducción a partir de "los nuevos bienes de consumo, de los nuevos métodos de producción y transporte, de los nuevos mercados, de las nuevas formas de organización industrial que crea la empresa capitalista" (pág. 121). En donde se destruye ininterrumpidamente la antigua y creando continuamente elementos nuevos. Donde existe una destrucción creadora que construye, lo que denomina "el dato de hecho esencial del capitalismo" (pág. 121) donde toda empresa que quiera sobrevivir debe adaptarse a esta dinámica.

En el caso de las empresas de software privado, el derecho de autor es una barrera que brinda protección suficiente para considerar invertir en desarrollo. Esto se debe a que les brinda exclusividad del desarrollo del software que han producido y hace artificialmente escaso el conocimiento, pensado como recurso que es objetivado. De forma paralela, esto abre la posibilidad para que un producto de software*,* que adquiere uso generalizado, se convierta en un estándar de *facto.* Esta característica ha favorecido el desarrollo del mercado del software[83], pero también, de manera semejante, el desarrollo de monopolios basados en software, lo que los ha convertido en el estándar de *facto.*

Rápidamente salta a la vista el caso de Microsoft, que con su sistema operativo ha establecido un plataforma de desarrollo que ofrece acceso a prácticamente el 90% de las computadoras de escritorio que existen conectadas a Internet (W3Counter, 2010). O bien, la existencia de Microsoft Office y su estándar de *facto*[84] de los documentos electrónicos, con lo que

[83] A través de la instauración de una plataforma generalizada para desarrollar *software*, como lo es el sistema operativo Microsoft Windows.

[84] El estándar de *facto* de la *suite* ofimática Microsoft Office, ha hecho posible que se puedan compartir archivos, generalmente de texto, hojas de cálculo y presentaciones de diapositivas, generados por los usuarios de las computadoras, pero también ha significado la necesidad de adquirir el programa si se desea tener acceso al contenido íntegro de los

obliga a cualquiera que desee competir con él a desarrollar compatibilidad con sus productos.

Esta compatibilidad, no puede realizarse observando el código fuente y se debe realizar a través de la experimentación, denominada "ingeniería inversa".[85] Aplicar dicha técnica de desarrollo, para los documentos de MsOffice, ha tomado años de trabajo y no ha podido presentar un código que ofrezca perfecta compatibilidad con estos formatos, ya que mucha de la tecnología necesaria para desplegar correctamente aquellos archivos es propiedad de Microsoft[86].

Ahora bien, no se trata sólo de Microsoft, siempre que existe un nuevo producto dominante en el mercado, trae aparejado la posibilidad de producir archivos que requieren el mismo programa para ser visualizado. Ya que el producto llamado software es de su propiedad intelectual, por lo que esta situación se puede ejemplificar con cualquier programa distribuido bajo licencias de uso, como es el caso de Photoshop, AutoCad, 3D Max, etc.[87]

Lo anterior sucede, debido a que el pago de una licencia de uso, ésta no concede la posibilidad de hacer modificaciones. O sea, niega al desarrollador de software el mismo conocimiento que el licenciatario tiene sobre el software, es decir, perpetúa la exclusividad sobre conocimiento formalizado. Esta implicación, trae como consecuencia que en cualquier desarrollo de software, sea necesaria la inversión tanto de capital como de conocimientos, que le permitan ser calificado como originales y, por lo tanto, cuente con la protección de derechos de autor.

El costo de desarrollo de software original puede variar mucho. Sin

archivos generados por este programa.

85 La "ingeniería inversa" es el proceso de descubrir los principios de funcionamiento de un dispositivo, objeto o sistema a través del análisis de su estructura, función y operación.

86 Puede observarse como un caso particular de esta situación el caso de la tecnología VML en Microsoft Office, tal como lo presenta Larson (2007).

87 Esta condición no deseada del mercado se ha naturalizado, al grado de que se ha invisibilizado. No obstante, esto se hace evidente si pensamos otras situaciones donde Microsoft no esté involucrado, por ejemplo: una aplicación que permita editar páginas para publicaciones profesionales. La primera que saltará a la vista es QuarkXPress, utilizado por 80% de los editores profesionales (Moseley, 2000), y que detenta un virtual duopolio seguida por Adobe In-design con una porción de mercado 8 veces menor hasta el 2006 (Walsh, 2006).

embargo, en el caso del software más complejo, como por ejemplo un sistema operativo, hoy en día requiere miles de millones de dólares. En el caso del sistema operativo de Windows Vista, de Microsoft, se afirma que han sido necesarios aproximadamente 6 000 millones de dólares para su desarrollarlo (Oiaga, 2007). El alto costo de la inversión que se requiere para producir un software de este tipo, limita el número de empresas privadas que estarían en condiciones de desarrollar otro. Asimismo, reduce la expectativa de que sea posible recuperar la inversión necesaria, bajo el modelo que requiere realizar ganancias a través de la venta de licencias.

El hecho de que la columna vertebral de la industria del software se haya construido en torno a la protección de conocimiento codificado ha tenido dos consecuencia, como ya mencionamos: a) el establecimiento de una industria del software, dado el dominio de un sistema operativo dominante que ha brindado una plataforma para la cual es posible desarrollar aplicaciones, y a la vez; b) una necesidad social, que se deriva de la expectativa del consumidor de poder compartir el producto del uso del software, es decir, los archivos (*out-puts*) que cualquier usuario haya creado con un software. Lo anterior, significa que todo aquel que requiera intercambiar datos producidos por un conocimiento codificado (software), debe tener capacidad de acceso a ese mismo conocimiento codificado (software) para consumar la puesta en común. Cuando la interoperabilidad[88] se encuentra construida en torno a conocimiento codificado protegidos por derechos de autor, entonces la empresa que detenta aquellos derechos sobre el software dominante -en algún nicho específico del mercado- tiene un monopolio.

Dadas estas restricciones, una empresa que tenga intenciones de entrar a algún nicho de mercado ya establecido, no solo requiere ser innovadora, requiere también: en primer lugar, la capacidad de sobrellevar la inversión de desarrollar software desde cero. Y en segundo lugar, ser capaz de crear usuarios de su software. Que si bien, lo último se puede sustentar en ciertas

[88] Por interoperabilidad se entiende “capacidad de los sistemas de tecnologías de la información y las comunicaciones (TIC), y de los procesos empresariales a los que apoyan, de intercambiar datos y posibilitar la puesta en común de información y conocimientos” (Parlamento Europeo, 2004, pág. 27).

innovaciones lo más importante es que aquellas, sean suficiente para crear o seducir usuarios para este software en particular. Si agregamos el componente de la interoperabilidad del software, entonces tenemos una barrera fundamentalmente social, sobre todo en la fase de introducción de un nuevo producto de software a un mercado ya establecido.

Esto último, plantea que en el contexto de una sociedad capitalista las circunstancias legales se encuentran diseñadas para impedir que innovaciones destruyan a las empresas ya establecida. Las empresas se pueden mantener en el mercado porque sus tecnologías tienen la característica de ser las más usadas aunque no necesariamente las más eficientes o innovadoras.

ii. El lock-in *tecnológico*

El *lock-in* existe en varias formas de producción que no hacen uso de formatos convencionales, tal es el caso de baterías, discos, cassettes, memorias, módulos, que sean desarrollados para su uso con algún artículo específico. En el ámbito del software, el *lock-in* se encuentra íntimamente ligado al apartado anterior, y se refiere al poder que tiene aquel que detenta los derechos de autor para decidir cuanto tiempo más actualizará el software o soportará determinadas versiones del mismo. Así, el futuro de la aplicación se encuentra atada a quien detenta los derechos de autor y no a aquel que ha comprado una licencia de uso. Puesto que, éstas pueden expirar después de cierto tiempo y no incluir derecho a nuevas actualizaciones.

Aquí, el que está a merced de la situación es el consumidor del software. Quien no tiene capacidad para modificarel software. Esto se puede hacer relevante cuando se requiera adaptarlo a otros sistemas operativos, o también, para adaptarlo a nuevo hardware.

Por otro lado, el *lock-in* es una ventaja para quien detenta los derechos de autor, pues es capaz de explotar la ventaja social que tiene el emplear el mismo producto que la mayoría de la gente. De esta manera, aquel producto que sea dominante en el mercado será aquel que sea más usado, lo cual será algo que dificulte el cambio a cualquier otra plataforma, pues tiende a

acostumbrar a los consumidores a su propias interfases y rutas operativas (Kavanagh, 2004).[89]

Por otro lado, en el caso de los desarrolladores de software, que desean hacer software para algún sistema operativo, requieren lo que se llama "Interfaz de Programación de Aplicaciones" (API; Application Programming Interface), la cual es un ambiente de programación que permite al creador de software pedir servicios al sistema operativo, esto con el objetivo de utilizar los recursos de este último. Y lo cual también puede ser usado para limitar el acceso al mismo (Kavanagh, 2004; Tanenbaum, 2003). Pero como ya se ha mencionado más arriba, no es la única forma en la que puede realizar *lock-in*, ya también es posible hacerlo desde el punto de vista del consumo, al hacer el *out-put* de un software en formatos propietarios, protegidos por derechos de autor Lo cual hace que todo producto de usar software requiera exactamente el mismo software decodificador[90] para ser leído por su autor.

iii. EEE: Adoptar, extender y extinguir

La estrategia que abordamos en este subapartado, es aplicable en la industria del software a protocolos o lenguajes que se considera apropiado controlar para ganar una posición dominante en un mercado dado. Estos lenguajes y protocolos, pueden ser libremente accesibles para los desarrolladores de software, con el objetivo de que sea acogida y utilizada para hacer nuevas aplicaciones.

Por consiguiente, cuando esta tecnología es adoptada, también es susceptible de ser extendida. Ahora bien, cuando dichas extensiones son hechas para funcionar únicamente con un software privativo, entonces se transforma en un problema si se constituye en una extensión es ampliamente utilizada. Que un sólo software pueda hacer uso de un protocolo o lenguaje con una extensión específica, es una dificultad cuando ésta se utiliza de manera

[89] En cambio, aquellos productos que se basan en estándares, como los servidores o navegadores de Internet no presentan grandes problemas para cambiar de sistema operativo (Windows-Linux y viceversa) o programa de administración para servidores (Windows-Apache y viceversa).

[90] Aunque no necesariamente la misma aplicación, sino sólo el software que lo pueda decodificar.

generalizada.

Sin duda alguna, se trata de un tipo de *lock-in*, pero tiene la característica que la hacen diferente. La más notable, es que no se trata el dueño de la tecnología el que decide que hacer con ella, sino aquel que tiene el control de la plataforma más usada. Esto se dio en varios casos protagonizados por Microsoft[91], sin embargo han habido otras compañías que también lo han utilizado. Tal es el caso de Netscape, cuando realizó extensiones al *hiper text markup language* (HTML), que no cumplían con los acordados por la World Wide Web Consortium[92].

Las tres estrategias, que hemos mencionado anteriormente, constituyen consecuencias concretas de la forma en que se protege el software en la sociedad capitalista del conocimiento. Y cómo éste favorece la capacidad de reproducción del capital, a través de permitir la concreción de la ganancia y a que proporciona las condiciones necesarias para su reproducción.

A continuación pasamos a describir el caso paradigmático de reproducción de la ganancia a través de licenciamiento de software privado. Microsoft fue la empresa que estableció las licencias de uso como el estándar que permitió construir la industria de software sobre su sistema operativo, y por ello la observamos.

c. Caso: Microsoft

El modelo de la empresa del software privado se ha vuelto dominante con el establecimiento paralelo de Microsoft como la empresa dominante del software. Esta empresa es un caso emblemático del desarrollo de software, porque hizo posible una plataforma común de desarrollo para los programadores y les ha permitido realizar ganancias a partir de la venta de licencias de uso. En el que la pieza central ha sido la forma en que se han establecido restricciones legales a la formalización del conocimiento codificado, que hace posible la monopolización y centralización de su desarrollo.

---

91 Que también le ha dado nombre a esta estrategia, a través de documentos oficiales que generó para uso interno de la compañía (Valloppillil, 1998).

92 Es una organización que se encarga de desarrollar estándares para la *world wide web.*

El modelo de desarrollo que ha impuesto Microsoft, tal como lo ha caracterizado Raymond (1998) es similar al de construcción de una catedral, donde es necesaria una estructura centralizada que controle y coordine el proceso de su construcción y en el caso del software, de su desarrollo.[93]

Con todo esto, la realización de ganancia en el software no se limita sólo a la forma en que se sintetiza el conocimiento en software de la manera más eficiente posible. También han de tomarse en cuenta las características que tiene el mercado cuando el producto que se comercia es conocimiento codificado privado, tal como hemos referido más arriba.

Para no extendernos en las diversas características que se pueden desprender de este caso, nos limitamos a atender únicamente el modelo que utiliza esta empresa para realizar valorización de software. Y en segundo lugar, las prácticas reales en el mercado en las que ha incurrido para lograr realizar ganancia.

Para lograr, en primer lugar la formalización de conocimiento tácito en diversos productos de software Microsoft ha formalizado su propia experiencia en lo que denomina como *Framework*[94]*.* Este último designa las guías para realizar el desarrollo y entrega de aplicaciones[95] y que denomina *Microsoft Solutions Framework* (MSF). Desoués, en la segunda parte de este apartado, abordamos de manera concreta la forma en que el legítimo monopolio del conocimiento codificado, le ha permitido a Microsoft hacerse de un mercado y mantenerlo.

i. Principios y modelos de valorización del conocimiento: Microsoft Solution Framework

El MSF fue introducido como tal en 1994, y presenta una colección de las mejores prácticas de Microsoft product groups, Microsoft Consulting, Microsoft's internal Operations and Technology Group (OTG), que se derivan de más de 30

93 Lo cual se aplicaría a desarrollos complejos, como es el caso de sistemas operativos.
94 El framework no hace mención de una metodología o modelo de desarrollo específico, y deja a los programadores decidirla.
95 Aunque también, esta empresa lo aplica al desarrollo de diversos proyectos en el campo de las tecnologías de la información e incluso en proyectos de infraestructura.

años de experiencia en el sector (Turner, 2006).

No hace énfasis en ninguna metodología de específica de desarrollo, sino en una serie de pautas que lo hacen más bien presentar una perspectiva. Cuando se utiliza el MSF, para desarrollar software, es más común que se utilicen las metodologías de cascada, espiral y *agile*[96], pueden ser aplicadas e incluso combinadas dependiendo del software que se trate de desarrollar.

Ahora bien, el MSF se basa en 8 principios fundacionales:

1. Fomenta una comunicación abierta;
2. Trabaja hacia una visión compartida;
3. Da poder a miembros del equipo;
4. Establece una rendición de cuentas clara y comparte responsabilidad;
5. Se encuentra enfocado en entregar valor de negocios;
6. Mantenerse ágil, esperar cambios;
7. Invertir en calidad;
8. Aprehender de todas las experiencias (Turner, 2006, pág. 21 T. del A.).

Aplicando estos principios, el MSF describe el papel de diversos miembros de un equipo de desarrollo de software de la siguiente manera:

1. Gerente del producto: Quien tiene la función de brindar una definición al producto, que satisface las necesidades y expectativas del inversionista o en su caso el consumidor.
2. Gerente del programa: Tiene la tarea de conseguir la solución de la inversión, lo cual óptimamente satisface las necesidades y expectativas del patrocinador(es) o en su caso el consumidor.
3. Arquitectura: Se encarga de asegurar que el diseño satisfactoriamente cumpla con todas las necesidades y expectativas.
4. Desarrollo: Esta tarea se divide en dos partes, la primera llamada “construcción de solución” se encarga de aquella que óptimamente satisface el diseño. Y la segunda, “Verificación de la solución” se encarga de revisar que la solución trabaje tal como se ha especificado.

[96] El modelo de cascada y espiral son formas centralizadas y jerárquicas de producción de software, en cuanto a la dirección que delinea una ruta a seguir del cómo ha de ser el resultado final del desarrollo. Sin embargo, estos modelos presentan el problema de que añadir gente a las unidades encargadas de desarrollar software aumenta su complejidad produciendo retrasos y propensión a fallas en la programación (*verbigratia:* ley de Brooks). El *agile* nace como una reacción en contra de los modelos de desarrollo inflexible, que no apuestan por revisar las anteriores fases de desarrollo (Beck & Beedle, 2001).

5. Pruebas: Se refiere a la labor de validar la solución, es decir, corrobora que la solución trabaje tal como se espera.
6. Experiencia del usuario: Tiene la doble función, de por un lado verificar que exista un efectiva, productiva y eficiente experiencia de usuario. Y por otro lado que la solución se encuentre lista para que los usuarios la aprovechen
7. Despliegue de la solución: Se refiere a cuando la solución se encuentra suavemente desplegada y óptimamente integrada en su(s) ambiente(s) objetivo(s) (Turner, 2006, pág. 47 T. del A.).

Donde a pesar de que este modelo, podría parecer como un organigrama, las tareas de desarrollo tienen más bien la función de ser abogados para cada uno de los objetivos que tiene cada papel. Así, se tiene la intensión de que se mantenga una estructura flexible, adaptable y escalable allende una persona ciertamente puede tener múltiples roles, aunque el propio MSF, tiene sugerencias sobre quienes y cuando se puede hacer esto.

Sin entrar en mucho detalle sobre la estructura organizacional que proporciona el MSF, podemos apreciar que se trata de un modelo que aún cuando aspira a ser mas flexible. Existen ciertas limitaciones propias de su estructura, ya que se mantienen islas con respecto a otros equipos de desarrollo de otro software.[97]

Tal como apunta Vinod Valloppillil (1998):

> (...) un desarrollador en Microsoft trabajando en el sistema operativo no puede ni arañar la superficie de Excel, como tampoco puede un desarrollador de Excel arañar la superficie del sistema operativo. Ya que les tomaría meses darse cuenta como construir y depurar e instalar, y además probablemente no podrían obtener acceso apropiado a los códigos fuentes de todas maneras (T. del A.).

La barrera del conocimiento, se encuentra incluso en el interior de la propia empresa que intenta desarrollarlo. No porque todos los programadores deban saber todo, sobre todo lo que produce la empresa, sino que si se necesita saber cualquier cosa en algún momento dado, sobre algún programa en particular, no es posible ni siquiera imaginar que se puede utilizar y/o aprender lo desarrollado en alguna otra parte de la misma empresa. De esta forma, cobra importancia la división de la accesibilidad al conocimiento, que supone este modelo privado de administrar la creación de conocimiento

97

codificado.

ii. Estrategias de realización de ganancia de Microsoft: Monopolio, lock-in *y EEE*

A marzo de 2010 Microsoft licencia su sistema operativo Windows 7 que cuenta con 11.90% del mercado, según la encuesta W3counter (2010). A pesar de ello, como su sistema operativo más usado aparece Windows XP, en el que corren 52.78% de las computadoras de escritorio de las más diversas marcas, seguido de Windows Vista (18.86%). Y le sigue con un distante 7.95% el Mac OS X de Apple.

Esta empresa, reportó 14 972 millones de dólares en ganancias en el 2007, únicamente por ventas de licencias de sistemas operativos para la computadora de escritorio (Microsoft, 2007a). Dados estos números, se puede calificar a Microsoft de ser un monopolio del sistema operativo dedicado al escritorio. Estas características del mercado le brindan a esta compañía la capacidad de realizar diversas acciones con él.[98] Entre las que se pueden incluir cualquier práctica propia de los monopolios, como sería *dumping*[99], para evitar que cualquier sistema operativo privado entre en este mismo terreno en una escala internacional. Ya que permite que desalentar la competencia e innovación, de empresas que se verían obligadas a competir con precios bajos de manera artificial.

Por otro lado, una característica más propia de este caso es la existencia de una gran cantidad de usuarios que ahora tienen conocimiento del sistema operativo Windows. Esta característica, hace aún más difícil que cualquier empresa capitalista tenga interés en invertir en un proyecto que debe entrar en un mercado ya ampliamente dominado por un sólo productor, y probablemente

98 No es de sorprender la existencia de diversas leyendas de tal marca “recomienda Windows” a pesar de que no exista una opción real para comprar el equipo con otro sistema operativo preinstalado que no sea el propio de la empresa Apple Inc.

99 La definición básica que nos ofrecen Bibek Debroy y Debashis Chakraborty (2007) de *dumping* es: “(...) exportar un producto a un precio más bajo que su «valor normal», mientras que el valor normal se define como el precio, «en circunstancias ordinarias de comercio, para un producto similar cuando se destina al consumo en el país exportador»” (pág. 17 T. del A.).

quede a la saga de la innovación que le permitiría realizar alguna destrucción creadora (recordando a Schumpeter).

En este sentido, se entiende el porqué, Jeff Raikes, presidente del Microsoft Business Group, comentó:

> Si ellos van a piratear a alguien, nosotros queremos que sea a nosotros en lugar de cualquier otro, [ya que] entendemos que a la larga, el recurso fundamental es la base conformada por la gente que tiene instalados nuestros programas y que los está usando. Lo que esperamos, a la larga, es convertirlos en licenciatarios del software (en Mondok, 2007)

La gran extensión de usuarios de los sistemas Windows y su gran poder financiero[100], hacen posible a Microsoft, bloquear la innovación al hacer improbables las posibilidades de éxito al realizar ganancias producto de la venta de licencias a los consumidores finales. De esta manera, esta compañía es capaz de realizar *lock-in* tecnológico de varias formas.

En primer lugar, desde el sistema operativo, donde el software más popular en diversas disciplinas se encuentra sólo para este sistema operativo. Pero también, a partir de las distintas aplicaciones que ha desarrollado, como es el caso de Microsoft Office[101] o Internet Explorer.

Como ya hemos mencionado más arriba de manera brevemente, el uso de Microsoft Office, es un estándar de *facto.* Dado su amplio uso permite a los usuarios realizar intercambio de archivos de manera confiable. Pero esto también, ha llevado a desarrolladores de otras empresas[102] y productos similares[103] a tratar de desarrollar compatibilidad a través de ingeniería inversa[104].

Del mismo modo, a Microsoft le ha interesado ganar control de nuevos

[100] En 2008 Microsoft logró ganancias netas de alrededor de 17 680 millones de dólares (Pingdom, 2009).

[101] El 90% de las ganancias de 16 396 millones de dólares que produjo en 2008 la Microsoft Business Division, proviene de la venta de Microsoft Office (Microsoft, 2007a).

[102] Como es el caso de Sun Microsystems, IBM, Novell, Corel, Apple Inc. Etc.

[103] Como programas de suite de oficina: OpenOffice.org, StarOffice, Lotus Symphony, Corel WordPerfect, etc. y varias otras aplicaciones buscan tener compatibilidad con los archivos que produce esta aplicación, como es el caso de Dataviz, Atlasti, RefWorks, MindManager, etc.

[104] Tal es el caso de el cambio de formato de almacenamiento que hizo Microsoft en su producto Microsoft Office. Al introducir Open Office XML, desarrollado para la versión 2007 y que no podía ser leído por versiones anteriores de Office y se publicó la especificación del formato para hacerlo un estándar internacional.

lenguajes y protocolos, a través de lo que se ha llamado *embrace extend and extinguish* (EEE). Un ejemplo de esto, es el que se dio entre los navegadores Internet Explorer y Netscape, al considerar que si perdía el control del navegador de Internet que se encontraba instalado en la mayoría de sus sistemas operativos perdería el control del propio sistema (Valloppillil, 1998).

En este caso, Netscape había desarrollado extensiones propietarias a protocolos ampliamente utilizados, por lo que Microsoft supuso que los programadores empezarían a desarrollar aplicaciones sólo para Netscape y no para la plataforma Windows y estándares que ella controlaba. Por lo que ésta hizo lo propio, y comenzó a liberar sus propias extensiones al lenguage HTTP, para de esta forma, obligar a los desarrolladores de sitios *web*, a hacerlos compatibles con su tecnología.[105]

Otro caso en el que se ha utilizado EEE, se puede observar en el caso de Java (un lenguaje de programación diseñado para ser ejecutado en varios sistemas operativos), que tras haber sido adoptado y extendido por Microsoft, estas mismas extensiones ya no le permitían ser compatible con el estándar que suponía interoperabilidad con otros sistemas. Quitándole su principal ventaja, y convirtiéndolo en la "ultima, mejor manera de escribir una aplicación para Windows" (Richtel, 1998).

En este contexto se entienden los "temores"[106] que compañías como Adobe tienen de que Microsoft realice extensiones de su estándar *Portable Document Format* (PDF). Que ha sido reconocido por la Organización Internacional de Estandarización (ISO), y que se ha hecho disponible a todos los desarrolladores de software en distintas aplicaciones.

En resumen, la posición dominante de Microsoft en el mercado, le ha facultado para ejercer prácticas monopólicas y le permite desalentar la adopción de tecnologías innovadoras que considere que amenazan su monopolio (ejerce poder de mercado). Donde al parecer, aunque el Estado pueda ser capaz de imponerle sanciones, no existe alternativa real que pueda provenir de esta

[105] Véase por ejemplo: La resolución que dio The United States Department of Justice en el caso Netscape v. Microsoft (1999).

[106] Véase por ejemplo: La nota, "Adobe Speaks Out on Microsoft PDF Battle" disponible en: CXO (2010).

estructura de creación de conocimiento y de esta forma de realización de ganancia.

A pesar de lo anterior, el sistema capitalista ha mostrado gran capacidad de adaptación. Las barreras de un monopolio con estas características, como es la legitima exclusividad del conocimiento codificado, han producido nuevas formas de producción del conocimiento y nuevas estrategias para realizar ganancia. Sobre todo, en nichos en los que es necesario compartir conocimiento.

2. Modelo de valorización del software libre

En este apartado abordamos el cómo se realiza la valorización en el software libre y cuales son las estrategias de realización de ganancia. Por ello, observaremos particularmente el cómo se han adaptado estrategias alrededor de un bien que se ha caracterizado como un “bien club”. Sin perder de vista que se trata de trabajo objetivado, susceptible de ser adaptado y al que se le pueden prestar servicios.

Así, comenzamos por explicar los distintos modelos de producción que giran en torno al software libre, y la manera particular en que se valoriza el conocimiento codificado, como un bien inmaterial o intangible. Posteriormente, se resaltarán algunas cuestiones históricas que hacen evidente como se han podido gestar las estrategias de su comercialización, abstraídas en modelos. Por último, presentamos las estrategias concretas que han tenido IBM, Red Hat y Novell, para con el software libre y la forma en que han podido realizar ganancia a partir de él.

a. Valorización del conocimiento

Eric S. Raymond en su famoso ensayo, “La Catedral y el Bazar” (1997), definió la diferencia entre el modo de desarrollo del software centralizado y el descentralizado a través de la analogía con la construcción de una catedral y la empresa privada y el modelo del software libre, con un bullicioso bazar.

En el primero, el modelo de catedral, describe que se caracterizaba por

estar dirigido por unos cuantos "(...) genios o pequeñas bandas de magos trabajando encerrados a piedra y lodo, sin liberar versiones beta antes de tiempo" (Raymond, 1998). Que se caracteriza por presentar a los errores y problemas desarrollo (*bugs*) como " (...) fenómenos truculentos, insidiosos y profundos" (Raymond, 1998), que toman meses para ser resueltos a través de revisiones exhaustivas. Para luego, darse cuenta de que no han sido resueltos cuando se presentan las versiones finales. Dicha situación, además, desmoraliza a los desarrolladores.

Por otro lado, contrasta el modelo anterior con el bazar, compuesto por varios individuos "(...) con propósitos y enfoques dispares (…)" (Raymond, 1998) que se encuentran dispuestos a cooperar en algún proyecto, porque saben que en este modelo de desarrollo cualquiera puede realizar aportaciones en algo que podrán utilizar. De esta manera, en este modelo los errores de desarrollo se hacen relativamente evidentes cuando se hacen públicos a miles de programadores que pueden realizar potencialmente miles de correcciones, lo que acarrearía un "efecto colateral benéfico[,] el perder menos cuando un eventual obstáculo se atraviesa" (Raymond, 1998).

La percepción de Raymond (1998, 2000) ha sido atendida por Weber (2000). Quién concuerda con el primer autor en que existe contribución que puede provenir de muchos lugares. Pero también toma en cuenta cuestiones económicas y sociales, que lo llevan a apuntar la importancia que tienen las estructuras de autoridad al interior de las comunidades del software[107] libre. Las cuales, son las que finalmente deciden qué contribuciones serán incorporadas en la siguiente versión del software.

Weber (2000) apunta que el proceso de desarrollo del software libre funciona cuando:

- Las contribuciones se basan en conocimiento que es ampliamente disponible.

[107] Cada proyecto de *software* libre tiene diferencias en los mecanismos que le permiten considerar y en su caso, aprobar, las aportaciones de la comunidad, sin embargo, muchas veces existen en estos posiciones privilegiadas para tomar decisiones, que se derivan de la confianza que se tiene en aquel que tiene el puesto de decisión sobre el proyecto. En el caso de Linux, por ejemplo, existen lugartenientes que se encuentran encargados del mantenimiento y supervisión de ciertos módulos del *kernel*, sin embargo la última palabra sobre el conjunto general y el rumbo que tiene el proyecto es de Linus Torvalds, quien actúa como un "dictador benevolente" (Weber, 2000; Fogel, 2007).

- Existe un “núcleo” que alberga la promesa de convertirse en algo bastante interesante.
- El resultado se percibe como importante, valioso, y de uso amplio.
- El resultado tiene una complejidad que puede ser desagregada en módulos paralelos.
- Y en el proceso de creación existen efectos positivos que se propagan en red (pág. 40 T. del A.).

Y el proyecto se desarrollaría mejor si:

- Los que contribuyen tienen confianza que sus esfuerzos generarán un producto y que no serán decepcionados.
- Aquellos que contribuyen valoran el *estatus* y reputación en parte como una recompensa simbólica, además del aspecto que le permite ser cuantificado como dinero.
- Los contribuyentes ganan conocimiento al contribuir al proyecto, cuando aprenden mientras hacen.
- El ego es un importante factor motivador.
- Aquellos que contribuyen piensan que se está haciendo algo “bueno” o “noble”, o que simplemente se encuentran oponiéndose a algo “maligno” (Weber, 2000, pág. 41 T. del A.).

Aunque, las conclusiones del análisis de Weber tiene que ver en gran parte con las motivaciones que impulsan la práctica en sí del software libre, lo que es importante notar es que el conocimiento que se comparte a través del software libre hace posible el proceso mismo. Es decir, en el proceso de transición del conocimiento de ser conocimiento tácito a sintetizar en el software conocimiento codificado explícito. Lo que tiene efectos que se propagan en red, pero sólo en ambientes que le permitan ser apropiado por tantos como tengan interés y capacidades cognitivas para hacerlo.

En este punto, es pertinente hacer notar que al parecer en el modelo de desarrollo del software libre ha sido potenciado por el Internet y a su estructura modular. Lo anterior, le ha permitido ser mejorado de forma paralela sin importar el lugar geográfico donde se encuentran los diversos programadores que contribuyen con él. Esto ha hecho posible que haya miles de personas que colaboran en un proyecto sin aumentar su complejidad. Situación que de acuerdo con la ley de Brooks, habría aumentado su tiempo de desarrollo de

manera geométrica, pero que ha sido mitigada dados los avances técnicos (como hemos visto en el Capítulo II, verbigracia: CVS, Subversion, Bitkeeper, GNU Arch o *git*).

Ahora bien, el producto de este proceso, el software libre, que se ha definido como conocimiento sintetizado, ahora puede ser pensado como un "bien club". Y que a través de su producción y su consumo, es capaz de generar ganancias monetarias, es decir, es posible realizar plusvalía a partir del conocimiento del software. Por lo tanto, el valor del software que es en sí conocimiento, sólo se puede realizar si se ha aprendido.

En el siguiente apartado abordamos los principios en los que se basa la realización de ganancia a partir y alrededor del software libre. Posteriormente, pasamos a observar cómo se presenta el software libre dentro de las empresas que lo han adoptado y desarrollado. Por último, veremos casos concretos en IBM, Red Hat y Novell.

b. Principios y modelos de realización de ganancia alrededor del software libre

Las empresas que realizan ganancia con el software libre pueden presentar básicamente dos modelos el "vendedor único de fuente abierta" (Dixon, 2009) y el "servicio/soporte de fuente abierta" (Aslett, 2008; Dixon, 2009). Cada uno de ellos tiene particularidades en cuanto a su relación con las comunidades que desarrollan el software y los clientes a los que les ofrecen el producto o bien, el soporte.

i. Vendedor único de fuente abierta

Siguiendo a Dixon (2009), podemos distinguir que al interior del modelo de vendedor único de fuente abierta (VUFA), podemos distinguir que en su interior existe un sistema de cambio entre dos grupos de consumidores. El primero compuesto por una comunidad que se encuentra alrededor de un proyecto de software libre. El segundo es el mercado dominante que se rige por ganancias económicas. En medio de ambos mercados, se encuentra la gestión del VUFA.

En el mercado dominante existen organizaciones que necesitan soporte, servicios y capacitación. Con los recursos recaudados por el VUFA, es posible contribuir a pagar por el trabajo realizado a ingenieros y gerentes de producto. Los cuales, realizan la labor de adecuar el software al mercado, donde la mayoría de las mejoras que son desarrolladas, termina como software libre y disponible para la comunidad.

Así, el otro grupo de consumidores llamado comunidad, se beneficia. Ya que puede recibir, por parte del VUFA mejoras al código abierto, al agregar perfeccionar el diseño, agregar funcionalidades, aumentar su estabilidad y proporcionar traducción y documentación, que también pueden ser utilizadas, estudiadas y modificadas por los miembros de la comunidad. Las mejoras que se hagan en aquellos rubros, atraen más clientes y alienta que el ciclo se perpetúe.

Siguiendo a Dixon (2009), se pueden distinguir tres partes del modelo:

1. En la primera la comunidad de software libre tiene acceso a software que puede ser usado para sus propios propósitos. El software podría tener más funcionalidades y recursos que si fuera un proyecto que no estuviese ligado a algún vendedor privado. Ya que se beneficia directamente de los ingresos que obtiene de los consumidores a través de la empresa.
2. Los clientes ganan mejor software a un mejor precio. Así, éstos se benefician de las comunidades de fuente abierta que son capaces de producir software de alta calidad.
3. La empresa gana al crecer, pero también al incrementar el aprecio de los dos grupos anteriores.

En este modelo, el software que es hecho accesible a los clientes a través de mecanismos de comercialización muy similares al software propietario. Aquí la empresa que lo comercializa, sólo tiene lazos con la comunidad a nivel de la ingeniería, donde existe desarrollo y verificación de calidad. De esta forma, la actividad de estas empresas se acercan en la Ingeniería y el manejo del producto, ya que ejercen la administración del

proyecto.

También, se nota que a diferencia del modelo propietario -que tiene un área encargada de la comercialización, soporte y servicios- en este modelo lo ingenieros tienen contacto continuo con el grupo de consumidores que está compuesto por los miembros de la comunidad del software libre. Esto tiene la consecuencia de que estos "clientes" se encuentran involucrados en crear "todo el producto" y no sólo una pieza del mismo.

Además de lo anterior, todo ese producto se ha creado para cubrir necesidades concretas del mercado. Dicho de otra manera, no crea necesidades, sólo las satisface. Por tanto, en este modelo, el principio que guía el desarrollo es satisfacer las necesidades de la comunidad, este es su objetivo principal y en la medida que se logra, la comunidad es mantenida y crece alrededor de un proyecto.

Ahora bien, también existen empresas que no tienen una relación tan estrecha con la comunidad de software libre. Sin embargo, tienen un nicho de mercado y han logrado posicionarse como empresas líderes en la industria. Éstas se dedican sobre todo al servicio y presentan características diferentes al modelo VUFA.

ii. Servicio/Soporte de fuente abierta

Siguiendo la abstracción que ha hecho Dixon (2009), de los modelo de realización de ganancia alrededor del software libre, Aslett (2008) propuso otro modelo parecido que se adaptara mejor a Empresas que distribuyen software de código abierto con programas de servicio y soporte. Así, este modelo que ha presentado Aslett (2008) y que ha sido reconocido por Dixon (2009) consiste en:

1. Una serie de desarrolladores centrales del proyecto de software libre. Que designan el diseño y el camino a seguir para lograrlo, que puede ser que no estén ligados a la empresa que utiliza el software que producen (*core team*; compuesto por un comité o un "dictador benevolente").
2. Una comunidad más amplia. Que se encarga de probar el software en casos reales, hacer revisiones al código, documentarlo, brindar

traducciones, sugerir y desarrollar características del software, crear y mantener foros de ayuda, arreglar *bugs,* probarlo en diferentes escalas y en diversas configuraciones de equipos, proporcionar un panorama de noticias que son relevantes en algún sentido para el desarrollo del software.

3. El proveedor de servicios. Que utiliza el software, desarrollado por los programadores centrales y la comunidad, para lograr ventas de soporte y servicios en el mercado, para lo cual es necesario realizar marketing[108]. Es posible que la empresa que proporciona el servicio desarrolle mejoras al software para que sea más adecuado a las necesidades del cliente.

Ahora bien, este modelo presenta una serie de ventajas y desventajas. En primer lugar, desde el punto de vista de las empresas no se tiene el control del futuro del proyecto central de software libre. No obstante, el software con el que se trabaja, podría requerir mucho más capital del que disponen o que están dispuestas a invertir bajo una metodología privada de desarrollo (valorización) y comercialización (realización de la ganancia).[109]

Por otra parte, es de notar que las empresas se encuentran obligadas a realizar mejoras al software y publicar dichas mejoras, sólo en el caso que hagan modificaciones directas al código fuente de los programas protegidos a través de la GPL o licencias similares que exijan que lo hagan. El “plus” que ofrecen las empresas de este tipo pueden ser también aplicaciones o *drivers*[110] desarrollados con licencias privadas que corren en o junto a software libre.

Las contribuciones de las empresas que practican este tipo de modelo, no únicamente lo pueden hacer a través de código. Si bien, este último puede reforzar su diseño, desarrollar nuevas características o arreglar *bugs.* Estas empresas también puede colaborar con la comunidad al mejorar documentación, brindar traducciones, ofrecer foros de discusión y soporte e incluso al hacer donaciones a fundaciones que respaldan el software libre.

---

[108] Entiéndase labores para convencer a los clientes de su capacidad de satisfacer necesidades y de realizar una ganancia.

[109] Tal es el caso de Linux y empresas como las que veremos a continuación, que le brindan soporte.

[110] Programa de computadora que permite al sistema operativo utilizar algún *hardware*.

Es necesario, recalcar que estos modelos tienen estos rasgos tan particulares debido a que responden a un contexto específico e histórico. En particular, el que se encuentra delimitado por el mercado dominado por una forma específica de restricción económico legal del conocimiento. En seguida, se señala cómo encaja la adopción de software libre en el marco de la sociedad capitalista presente.

c. El software libre es una estrategia

Para entender la incorporación del software libre en el ámbito de las empresas capitalistas se propone entenderla como una estrategia. De esta manera, será posible dilucidar la forma en la que se hace la creación de conocimiento codificado avalado por un sistema económico legal. Lo que paralelamente, permite a grandes empresas -como las que se tratarán en el siguiente apartado- obtener ventajas en el mercado al socavar la porción de mercado que han controlado otras empresas dominantes. Esto último, se hace evidente si prestamos atención a distintos casos de software libre, a sus patrocinadores y se contrasta con su semejante propietario, tal y como presentamos a continuación.

En el caso del desarrollo de sistemas operativos, una distribución de GNU/Linux, se estima que hubiese costado 10 800 millones de dólares en 2008, sólo en investigación y desarrollo utilizando el entandar industrial: Modelo de Costo Constructivo (McPherson, Proffitt, & Hale-Evans, 2008). Si comparamos estos números con, los 6 000 millones de dólares que ha costado el desarrollo de Windows Vista (Oiaga, 2007), tenemos una idea de que se necesitaría ser, en primer lugar, una empresa de la envergadura de Microsoft que espera obtener ganancias netas de 17 680 millones de dólares al año (Pingdom, 2009) y/o esperar obtener en un año por concepto de ventas de licencias de sistemas operativos de escritorio 14 972 millones de dólares (Microsoft, 2007a). Nichos que ya se encuentran ocupados y tiene las ventajas estratégicas de ser un monopolio que es conocido por tener prácticas como *lock-in* tecnológico propias de ser un estándar de *facto* o bien realizar estrategias intencionales de cooptar

lenguajes y protocolos que son estándares reconocidos por diversas asociaciones.[111]

Del mismo modo, podemos reconocer a través de estudios estadísticos la clara rivalidad que tienen los productos de Microsoft en el sector de servidores. En enero de 2009, Netcraft estimaba a partir de una muestra de 56 323 sitios consultados, por grupo de sistema operativo instalado que Windows contaba con el primer lugar con 22 827 que equivalía al 40.53% de la muestra, seguido de cerca por Linux con 22 636 equivalente al 40.19% (Netcraft, 2009). Por otro lado, se estimaba que a Marzo de 2009, de los 206 675 938 sitios de servidores más ocupados un 54.55% de ellos utilizan el software libre de administración Apache, al sumar 112 747 166 unidades y dejar en un lejano segundo puesto a Microsoft con 24.47% al sumar 50 928 226 servidores (Netcraft, 2010).

Lo anterior hace evidente que en el mercado de sistema operativo y administración de servidores el rival a vencer es software Libre. Esto explica la urgencia que tenía Microsoft por atender el sector Internet con políticas como la EEE (Valloppillil, 1998), para encargarse de controlar los estándares que hacen posible la interoperabilidad con cualquier sistema operativo y lograr *lock-in* en el mercado de servidores, cosa que evidentemente aún no ha conseguido.

Este modelo de producción descentralizado, es entonces muy exitoso entre los usuarios con más conocimientos. Usuarios que se desempeñan en sectores informáticos donde tienen gran injerencia estándares informáticos y existe la posibilidad de hacer contribuciones que se basan en conocimiento ampliamente disponible (Weber, 2000). Ahí, los usuarios que cuentan con altos conocimientos, pueden salvar cualquier dificultad técnica que se les presente, ya que ese conocimiento les brinda control sobre el software.

En cambio, el uso del software libre en las computadoras personales de escritorio es más limitado. De una muestra de 34 286 de visitas a los sitios que participaron del estudio, sólo 32.1% lo hace con Firefox, o bien 6.8% con Chrome[112], ambos navegadores con componentes abiertos. Y superados por

[111] Como ya vimos más arriba en el caso de Java y el HTML.
[112] Navegador construido a base de código de fuente abierta por Google y la comunidad de

Internet Explorer de Microsoft con 48.7%(W3Counter, 2010). Del mismo modo, el uso del sistema operativo GNU/Linux en las computadoras de escritorio es mucho más limitado, sólo el 1.55% de los clientes de servidores evaluados por W3Counter (2010), lo tienen instalado. Esto puede deberse a que hoy en día casi el total de las computadoras domésticas se venden con Windows preinstalado.[113]

Sin embargo, la aparición de nuevas computadoras de capacidades técnicas limitadas para los estándares de hoy, llamadas NetBooks, han abierto la posibilidad de presentar Linux preinstalado al mercado[114]. La misma aparición de las NetBooks con procesadores de bajo rendimiento[115] es la que ha auspiciado la extensión del periodo de ventas del sistema operativo Windows XP. Que se retiró del licenciamiento general el 30 de Junio de 2008, pero se conservó sólo para obtener ganancias del creciente mercado de este tipo de computadoras hasta el 22 de Octubre de 2010 (Dix, 2008). En la fecha anterior, se espera que la mayoría de este tipo de computadoras cuente con suficiente memoria y procesador capaz de correr los sistemas operativos más recientes de Microsoft.

En este sentido, podemos observar cómo es que la existencia de software libre en el mercado permite a empresas, tener acceso a software que tiene capacidad de generar ganancias entre el costo de su aplicación y su rendimiento. Ya sea como un sistema dominante en el mercado de servidores o bien como oportunidad que permite cierta capacidad de intervenir en un mercado que se encuentra dominado por un monopolio.

Esto explicaría el porqué, empresas como IBM, Red Hat y Novell, han invertido en el desarrollo del *kernel* Linux (véase: Capítulo II), ya que a pesar de llegar a ser competidoras en ciertos sectores, han realizado mayores contribuciones a Linux en código y donaciones a la fundación Linux, ya que

Chromium.

113 Se trata de una situación de monopolio, a la que se le ha llamado incluso el “impuesto Microsoft” (Microsoft Tax), véase: Coulais (2009).

114 Donde según Jeff Orr, un analista de ABI, afirmó que en aquél momento (Noviembre 2009) 32% de las netbooks corrían alguna versión de Linux (Lai, 2009).

115 Varias netbooks operan con procesadores Intel ATOM, de bajo rendimiento y consumo de energía, al igual que los procesadores de la familia ARM.

realizar estas prácticas tiene la capacidad de descentralizar los riesgos de invertir en software.

OpenOffice.org y MySQ son otros dos ejemplos, que puede ser mencionados de manera sucinta. Donde el código base del primero fue liberado por Sun Microsystems de su producto StarOffice, para competir de manera directa con Microsoft Office[116], apelando a la colaboración de la comunidad del software libre. Hoy en día, el proyecto es apadrinado por Oracle Corporation, empresa que adquirió a Sun Microsystems, y a las que se han sumado contribuciones hechas por empresas específicas como IBM, RedHat Novell, RedFlag CH2000 y Google entre otros (Oracle, 2010). O bien, el caso de MySQL, sistema de bases de datos relacional, liberado como software libre, ahora también patrocinado por Oracle, que compite con otras aplicaciones como Microsoft SQL Server e incluso, con otros productos de la propia Oracle.[117]

Otro ejemplo que no se puede dejar de mencionar, es el desarrollo del sistema operativo Android. Basado en el *kernel* Linux y librerías GNU, y optimizado para su empleo en dispositivos de bajo rendimiento (celulares, netbooks y pads). Diseñado originalmente por Android Inc., ahora propiedad de Google, fue liberado el 21 de octubre de 2008 y en la actualidad es desarrollado por la Open Handset Alliance.[118] Dicho sistema, fue creado para competir directamente con el sistema operativo del iphone (iOS) de Apple en cuanto a su

[116] En el estudio hecho por Thomas Hümmer (2010) se muestra que en ciertos países europeos, el volumen de computadoras que tienen instalado OpenOffice (o programas con el mismo código base) alcanzan hasta 22%. Entre ellos, podemos mencionar: Polonia y República Checa con 22%, Alemania con 21%, Francia con 19%, Italia y Noruega con 18%, España con 15% y Bélgica y Suecia con 14%. Aunque el autor reconoce que es posible un margen de error de ± 5%, este estudio es suficiente para proporcionar una idea de la magnitud de adopción que hay en cada país.

[117] Según asevera Connie Sanchez (2007), un estudio realizado por Evans Data Corporation reveló que MySQL, ganó 25% del mercado que conforman todas las bases de datos utilizadas por desarrolladores en los dos años anteriores. Y donde el uso de MySQL, se proyecta que seguiría creciendo en el futuro, dado que los desarrolladores cada vez más usan fuente abierta. De acuerdo a esta misma fuente, el 65% de los desarrolladores lo utilizaba hasta el otoño de 2006.

[118] Compuesta por 65 miembros y cuyo número ha continuado en aumento. Entre los que destacan compañías de manufactura de dispositivos electrónicos, desarrolladoras de software y prestadores de servicios en telecomunicaciones, tales como: Acer, Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia, Wind River Systems y Vodafone (OHA, 2010).

sistema de interfase sensible al tacto y por el mercado de aplicaciones. Según un estudio de NPD Group presentado por Dan Gallagher (2010), la venta de dispositivos con Android fue del 28% del mercado en los Estados Unidos para marzo de 2010, superando al propio iphone, con 21% y cuyas ventas se mantuvieron estables desde diciembre de 2009.[119]

De esta forma, invertir en software libre, por parte de empresas grandes, se perfila como la mejor forma de minimizar riesgos. Sobre todo, a la hora de alentar la competencia de quien detenta monopolios (como es en el sistema operativo Microsoft Windows y en las aplicaciones ofimáticas Microsoft Office) en la industria del software. Donde Microsoft cuanta con buena parte del mercado en diversos sectores de la industria del software.

Así, estas compañías se encuentran en posición de evitar las desventajas de invertir en software dedicado a competir por un nicho de mercado ya ocupado, con usuarios establecidos, al descentralizar la producción geográfica y financieramente. Al contrario de las empresas privadas, el capital que se invierte es aportado por todo aquel que suma con su trabajo y conocimiento, formalizado en código u objetivado en forma de donaciones. Aquí, se cuenta con la certidumbre de que: si alguno de los contribuidores deja el mercado, el producto no necesariamente desaparece, ya que si cuenta con una comunidad robusta de programadores y usuarios, éstos pueden continuar con su desarrollo.[120]

---

[119] El líder del mercado es el sistema operativo del Black Berry, con 36% de ventas en el mercado estadounidense (Gallagher, 2010). Ahora bien, Microsoft aquí se encuentra en cuarto lugar, con el Windows Mobile, que se perfila para mantenerse en esa posición para el año 2012, según apunta Gartner (En Hamblen, 2009).

[120] El ejemplo más notable de esto, ha sido el software Blender, utilizado para construir y animar entornos y personajes detallados en tres dimensiones, los cuales, pueden utilizarse en series y películas animadas, juegos electrónicos e imágenes. Esta aplicación fue desarrollada por Not a Number Technologies (NaN), empresa alemana fundada por Ton Roosendaal en 1998 bajo un esquema privativo. En el año 2002 NaN se declaró en bancarrota, y Roosendaal acordó con sus acreedores liberar el código fuente de Blender bajo licencia GPL a cambio de un pago único de 100 000 Euros. El 18 Julio de ese año comenzó la campaña para reunir los fondos necesarios apelando a la comunidad de usuarios en particular y del software libre en general. Para el 7 de septiembre, se informó que se habían reunido los fondos necesarios y el código sería liberado. Hoy, Blender se encuentra en desarrollo activo, bajo la supervisión de la Fundación Blender (2009).

d. Casos

En este apartado retomamos de manera breve tres casos de la industria del software. Que son encarnados por empresas grandes que se han dedicado ha la comercialización de soluciones de software libre. Y en ellas nos concentramos en los modelos que han seguido para realizar ganancia a partir del software libre.

i. IBM

Como hemos visto más arriba, esta empresa ha estado implicada en la historia del desarrollo de la industria informática y en particular del software. Es una empresa que aunque hoy en día ha sido desplazada por Microsoft, como empresa dominante en la industria del software (dado que no tiene el control del sistema operativo más ampliamente usado), sigue siendo una empresa con gran influencia en el sector de tecnologías de la informático en general. Pues, en 2008 generó ventas por alrededor de 103 630 millones de dólares, aunque sus ganancias netas fueron de 12 340 millones de dólares. Esto contrasta comparativamente con Microsoft, quien tuvo 61 100 millones de dólares en ventas y obtuvo 17 680 millones en ganancias netas (Pingdom, 2009).

Esta empresa, ha reportado ganancias de 1 500 millones de dólares en ventas de servicios en 2001, relativos al software libre y expertos esperaban más de 3 500 millones para 2004 (Kavanagh, 2004, pág. 45). Para 2005 IBM tenía el primer lugar en el las ganancias derivadas servidores con Linux instalado. Con el 29.7% de las ganancias a nivel mundial (IBM, 2005). Donde Red Hat y Novell han jugado un papel esencial, al permitirle a IBM poder ofrecer servicios a distribuciones Linux específicas, como las que ofrecen aquellas compañías.

De esta manera, es de esperarse que esta empresa pueda presumir de haber invertido en Linux desde 1999, y con ello haber podido desarrollar:

- Compatibilidad para todos los servidores que produce
- Más de quinientos productos que corren en Linux
- Una línea completa de para poner en servicio soluciones específicas, dar soporte y

proveer servicios de migración a Linux (IBM, 2009 T. del A.).

Donde además, asegura que tiene trabajando a más de 600 personas en desarrollar alrededor de 150 proyectos de software libre en los que IBM activamente participa (IBM, 2009).

Entre las soluciones que IBM presume, a través del uso de Linux se encuentran

- Consolidación de servidores
- Clusters
- Empresas distribuidas
- Soluciones
  - En aplicaciones.
  - En infraestructura (Murphy, 2003).

De este modo, la propia IBM reconoce que se trata de un cambio de paradigma, donde su modelo de negocio no se encuentra en la venta del software, sino en la venta del hardware y software asociado que requiere servicios, el cual es un negocio que tiene ventas por miles de millones y sigue subiendo (IBM, 2005).

Con esto, vemos que IBM se acerca más aún al modelo VUFA, ya que tiene proyectos propios en los que invierte gran cantidad de recursos y que tiene el objetivo de ayudarle en las soluciones que ofrecen en el mercado. Aquí, esta empresa, hace uso del conocimiento que ha sido objetivado por las comunidades, y ha sido capaz de intermediar con el público consumidor. Al tiempo que las comunidades se han consolidado al rededor de sus productos.

Entre sus productos más populares, se encuentra el proyecto libre Eclipse. Pero también la llamada Websphare software de licencia privada. Este último, que se encuentra asociado a los servicios y soluciones de IBM que se ofrecen sobre la plataforma Linux. De esta forma, es posible vislumbrar la aparición estrategias que incluyen componentes propietarios y libres, permitiéndole a esta empresa, aprovechar software de alta calidad producido por la comunidad y presentarlo como un bien de consumo atractivos al cliente final, al tiempo que controla siertos aspectos del software a través de licenciamientos privados

ii. Red Hat

Es la empresa más grande que sólo presta servicio a Linux. Reportó ventas por 652.57 millones de dólares y ganancias netas por 78.72 millones en 2009 (Red Hat, 2009a). Se trata de una empresa relativamente nueva en la industria del software, fundada en 1993 con sede en Raleigh, Carolina del Norte. Tiene 2 500 empleados distribuidos en 58 oficinas satélite divididas en 28 países alrededor del mundo (Red Hat, 2010).

Ésta provee su propia distribución de GNU/Linux diseñada para empresas, llamada Red Hat Enterprise Linux. Asimismo, apadrina una versión sin el soporte propio de la versión empresarial y la cual, se encuentra disponible de manera gratuita en Internet, bajo el nombre Fedora Core. Esta última distribución, le permite probar nuevos paquetes de software entre los usuarios de la comunidad. Con ello, puede perfeccionarlos y seleccionar los que posteriormente serán incluidos en la versión estable que ofrece a las empresas.

De manera comprensible, todo componente de Red Hat Linux se encuentra licenciado bajo la GPL, a excepción de ciertos *drivers* y componentes desarrollados por terceros. De igual manera, incluso los componentes que ha diseñado Red Hat se encuentran disponibles bajo aquella licencia. Pero la integridad de su diseño se encuentra restringida por leyes de marca (*branding*), que prohíben a cualquiera hacer una copia exacta del software de esta empresa, pues se encontrarían copiando su marca y logos sin permiso, lo que viola el acuerdo de uso con el usuario (Red Hat, 2009b).[121]

Así, el modelo de realización de ganancia de Red Hat se encuentra en ofrecer software que se ha consolidado a través de un proceso que resulta atractivo para el cliente. Es decir, ofrece la garantía de que ha sido probado en situaciones reales y además la seguridad de que si se presentan inconvenientes, se puede recurrir a esta empresa por el soporte.[122] También se

[121] Dado que Red Hat publica sus modificaciones como software libre, existen versiones de este software que son distribuidas sin las marcas de Red Hat, pero no reciben soporte. Tal es el caso del Community ENTerprise Operating System (CentOS) que es integrado a partir de la reconstrucción de los paquetes de Red Hat Enterprise Linux.

[122] Red Hat ofrece paquetes de soporte que en 2010 van de 349 dólares en suscripción básica a 2 499 dólares en suscripción *premium* a su plataforma más avanzada.

beneficia de dar capacitación y servicios de consultoría a los clientes a una escala global, concentrándose en la seguridad y el fácil manejo (Red Hat, 2010).

Red Hat, también ha impulsado de manera consistente a la comunidad del proyecto Jboss. Desarrollado en Java, se trata de un software que originalmente era producido por Jboss Inc. Y que posteriormente fue adquirida por Red Hat y liberado como software libre bajo la licencia GPL. Este proyecto permite agregar ciertas funcionalidades a Apache-Tomcat para crear aplicaciones java en red.

Esta empresa se ha aliado a IBM desde hace 10 años y brinda soporte a sus sistemas *Mainframe* z10. Ya que según sostiene, ofrece un mejor control del sistema operativo, confiabilidad, disponibilidad, seguridad y procesos de negocios ya establecidos (IBM, 2008).

Lo anterior, nos sirve para ver cómo es que Red Hat se acerca más al modelo de Soporte/servicio. Puesto que, alrededor de él es que consigue mayor parte de sus ingresos. También, podemos observar que si bien tiene una distribución de GNU/Linux orientada a la comunidad, el desarrollo central de GNU y de Linux se encuentra ajeno a las decisiones que pudiera tomar Red Hat, sobre la orientación de su propio proyecto.

En este sentido, Red Hat es capaz de realizar ganancia a partir del software caracterizado como conocimiento codificado, o sea como un "bien club". Esto sucede cuando capacita sobre su uso, o sea, facilita su apropiación como conocimiento. Ya sea a través de certificaciones o de resolver problemas, problemas que quien los padece no puede solucionar por falta de conocimientos.[123]

A pesar de lo anterior, al igual que con IBM, es posible advertir un componente privado en su estrategia. Encarnado en la protección que tiene de la integridad de su producto a través de leyes de marcas, pero que no dejan de estar disponibles bajo la GPL que existe en su distribución GNU/Linux. Pieza

[123] Brindar soporte es a fin de cuentas: brindar el conocimiento necesario para resolver una situación contingente. Una vez que la contingencia es resuelta, la manera de resolverla pasa a formar parte del acervo de conocimientos a mano de quien la ha padecido.

que lo hace atractivo para el consumidor, que obtiene soporte y acceso directo a los paquetes “originales” que se encuentran directamente en sus servidores.

iii. Novell

En contraste con Red Hat, Novell es una empresa que tiene más años en la industria del software, fundada como empresa de hardware en 1979, entró en el negocio del software en 1983 con el *network operating system* (NOS) llamado ShareNet. En 1993 compró UNIX (Kavanagh, 2004, pág. 9) aunque después de un para de años vendió algunos derechos a Santa Cruz Operation (SCO)[124], y empezó a ser desplazado por Microsoft en el mercado de software. A pesar de ello, Novell sigue siendo una empresa grande en la Industria, tiene una total de $862 millones de dólares en ventas y un ingreso neto de $213 millones de dólares (Bednarz, 2010).

En 2003, Novell entró al mercado de GNU/Linux con el lanzamiento al mercado de la distribución SUSE Linux Enterprise. Que incluye compatibilidad con las tecnologías que ha desarrollado y mantenido de manera privada. Tal es el caso de NetWare ahora llamado Open Enterprise Server.

Hoy en día, sus negocios se centran en el soporte técnico por suscripción, pero también se ha concentrado en mantener comunidades de software libre a través de foros de ayuda técnica en línea así como Wikis. E incluso, algunos de sus productos incluyen información que ha sido producto de la retroalimentación que han recibido en foros.

Al igual que Red Hat, tiene una distribución GNU/Linux llamada Open Suse que no cuenta con la característica estabilidad de la versión empresarial ni con soporte o actualizaciones de seguridad desde sus repositorios. Pero que se encuentra disponible para la comunidad de manera gratuita y cuenta con los últimos paquetes disponibles.

Por otro lado, Novell y Microsoft han consumado un acuerdo mutuo para desarrolla e incrementar “(...) la interopaerabilidad de Linux (específicamente

[124] Existe disputa sobre qué derechos ha vendido Novell a SCO, donde al parecer se ha decidido que la integridad de los derechos pertenecen aún a la primera compañía (Jones & Erwan, 2010).

SUSE Linux Enterprise) y Windows y para «proveer a sus clientes con cobertura de patentes para sus respectivos productos»" (Deek & McHugh, 2007, pág. 231). Esto le posibilita ofrecer versiones de GNU/Linux compatibles con la tecnología que ha desarrollado Microsoft -como ".Net"-, así como versiones de OpenOffice, que tienen mucho mejor compatibilidad con Microsoft Office (Novell, 2010a, 2010b).

En la actualidad, Novell presume que Suse Linux Enterprise, es la mejor opción para realizar computación mixta entre Windows y Linux (Novell, 2010a). Esto le permite a Novell tener ventajas competitivas, ya que al igual que Red Hat, aquella empresa tiene acuerdos con IBM, y ofrece soluciones para el "sistema Z" (Novell, 2010c).

Con lo anterior podemos ver que, el modelo de desarrollo de Novell se ha movido de ser una empresa dedicada al desarrollo de software propietario a ser una empresa orientada a dar soporte y servicio. Así como ha destinado recursos a desarrollar software que brinde servicios de interoperabilidad con software libre, por lo que resulta ser más cercana a Red Hat en su modelo de Servicio y Soporte.

En este sentido, el componente que le brinda ventaja es la capacidad de ofrecer tecnologías que ha desarrollado y mantenido de manera privada, en plataformas abiertas y que son ampliamente usadas. También el ofrecer acceso a tecnologías que sólo se encuentran disponibles desde este vendedor en las plataformas abiertas, como las que ha acordado con Microsoft. Estas dos "ventajas" le permiten ofrecer recursos que sólo esta empresa controla a sus clientes, pues es capaz de restringir su apropiación como conocimiento. Por otro lado, es posible afirma que valoriza el conocimiento a partir de las dos dimensiones del software, únicamente como conocimiento objetivado cuando lo hace a partir de software privativo y de un "bien club" (conocimiento codificado) cuando hace contribuciones al software libre o brinda soporte.

Conclusiones

A lo largo de este capítulo hemos visto que la industria del software se ha

consolidado en torno a las restricciones al conocimiento formal codificado que hacen legítima su apropiación privada. Dichas restricciones, también han intervenido en la generación de conocimiento útil para el desarrollo del software, al igual que en la forma en que se han realizado ganancias a partir de su comercialización.

Como hemos observado, las empresas privadas del software aspiran a administrar el proceso creativo de codificar conocimiento y formalizarlo en software. Para ello se han intentado aplicar modelos rígidos de desarrollo que asemejaban la estructura industrial de la cadena productiva. Y el cual, ha demostrado ser ineficiente a la hora de tratarse de desarrollos que requieren gran complejidad y un gran número de personas comunicándose, lo que implica incremento en el presupuesto de desarrollo, retrasos e incremento en la propensión a *bugs.*

En las situaciones anteriores, se han marcado las dificultades que surgen al añadir personas a un proceso de creación de conocimiento (ley de Brooks). Pero también que éstas se han atendido a través del diseños técnicos modulares del software. Que permiten a varios grupos de trabajo realizar modificaciones sobre el mismo software sin obstaculizarse mutuamente.

A pesar de esto, ambas estrategias de producción de software, en la empresa privada, se caracterizan por tener como eje rector, a saber: la restricción del conocimiento. Por un lado, los grupos de programación al dividir el conocimiento sobre el software en diversas tareas, cuando se parcela el conocimiento del software sólo a los módulos asignados. En segundo, la apropiación del conocimiento por parte del capital, a través de su proceso de valorización, que siempre es jerárquica.

Ahora bien, el hecho de que no se trate de un bien material o tangible, ha producido la posibilidad de que exista la practica social del software libre y que haya sido potenciada por la revolución de las comunicaciones. Esta situación, ha logrado hacer evidente que es provechoso compartir el software que ha sido desarrollado para atender necesidades puntuales o que promete convertirse en algo interesante. Esto se debe a que aquel software, también puede ser

aprovechado e incluso mejorado por alguien más que sea capas de apropiarlo como un “bien club”, es decir, que tiene los conocimientos suficientes para estudiarlo, modificarlo y redistribuirlo.

Donde al parecer, los modelos de desarrollo de software libre, como el que se ha puesto en práctica en Linux o Apache, parecen haber superado el problema de incrementar la complejidad y el número de personas trabajando en él. Lo anterior, a través de la creación de estructuras de desarrollo y medios técnicos que logran desagregar la complejidad de los programas al dividirlo en módulos y permitir trabajar de manera paralela en cada uno de ellos, por tantos desarrolladores como puedan y quieran hacerlo. A la vez, esto permite la construcción de una librería de software que permite acceso a cualquiera de los módulos de los proyectos que se han creado, y su reutilización por cualquiera que los considere conveniente, ya sea dentro del proyecto que los ha visto nacer o en otro, cuyo valor de uso sea totalmente distinto.

Esta forma de producción de conocimiento codificado ha generado innovación. La cual, compite y ha logrado hacerse de una porción considerable del mercado en el que ya se encontraba establecido un vendedor único de software privado, Microsoft. Donde dicha empresa, cuenta con estrategias de diversos tipos como *lock-in* y EEE, que le permiten desalentar la competencia generada desde el ámbito privado para conservar su monopolio. A pesar de ello, éstas se han mostrado como insuficientes para contener el avance de software que ha sido desarrollado de manera libre.

Utilizar el software libre como estrategia subsumida en empresas privadas ha permitido que en el nicho de servidores el software libre se erija como software dominante. Como es el caso de Apache y de acuerdo a las estadísticas muy pronto Linux lo puede llegar a ser. Esto es posible, pues en este ámbito del sector se depende de estándares abiertos y existe conocimiento abundante sobre el tema. Así, cualquiera que cuente con conocimientos suficientes puede modificar el código y satisfacer sus propias necesidades. Y de paso, incluso aportar algo a la comunidad[125] que le ha brindado acceso al

[125] Aunque dadas las jerarquías al interior de las comunidades del software, éstas podrían no ser tomadas en cuenta.

conocimiento codificado.

Por otra parte, empresas privadas como IBM, Red Hat y Novell han invertido en desarrollos que hicieron más ricas las comunidades del software libre. De la misma manera, han desarrollado aplicaciones restringidas que han hecho más atractivas las soluciones basadas en código abierto a los clientes empresariales. Donde la estrategia de integración de software libre les ha brindado acceso a programas de alta calidad sin correr los riesgos de emprender un desarrollo privado en solitario.[126]

El conocimiento del funcionamiento del software libre, ha permitido consolidar modelos de comercialización que permiten ser lucrativos en una industria en la que parece dominar la estrategia de licenciamiento y con ella el monopolio legítimo del conocimiento. Dicho conocimiento, permite cristalizar valor de uso estable para el software libre, por lo que el soporte, la venta de soluciones y la capacitación de los usuarios es importante.[127] Así, el software libre se presenta como una estrategia que ha permitido a empresas grandes mantenerse e incluso crecer en un mercado que se encontraba aparentemente dominado por un monopolio invencible (*verbigratia*: el sistema operativo y las suites ofimáticas).

Tenemos pues, que existen en la sociedad del conocimiento capitalista, un contexto inmediato que presenta dos tipos de empresas. El primer tipo, se encuentra constituido por empresas que tienen el monopolio del conocimiento codificado. El segundo tipo, por empresas que hacen uso de software que no se encuentra sujeto por restricciones económico-legales que legitimen no compartirlo.

Ante esta realidad, es posible identificar aquellas que emplean las ventajas de este software que es de libre acceso y que les permite, generar nuevo conocimiento o bien adaptar el existente. De esta suerte, las nuevas formas de valorización del conocimiento, que se distinguen por la práctica de

[126] El caso de Linux es paradigmático, en especial cuando se considera a la luz del más reciente ejemplo que presenta Android. En el que una alianza de más de 60 empresas se han unido para crear dicho sistema operativo y que, como hemos visto, hasta el momento muestra crecimiento sostenido frente a productores únicos.

[127] Bien que, tratarse de sistemas parecidos a UNIX, la curva de aprendizaje resulta mucho menor para los usuarios ya iniciados en esta materia.

compartirlo, es adoptada en el ámbito empresarial para lograr realizar ganancia. Esto ante un mercado que presenta nichos de mercado que presenta altos riesgos si se sigue únicamente la lógica de desarrollo y licenciamiento del software privado.

En el siguiente capítulo abordamos el cómo se presenta el software libre en las empresas que conforma la AMESOL. Y cual es la realidad en que lo usan, lo valorizan y es posible hacer ganancia de un "bien club". Asimismo abordamos como su estrategia también ha abierto un mercado que sin el componente libre, potenciado por las nuevas tecnologías de la información, se había presentado como inaccesible a cambios y cerrado, este último producto del legítimo monopolio del conocimiento formalizado. Por último, en el siguiente capítulo, también será posible seguir el desenvolvimiento histórico de las empresas capitalistas y el doble movimiento que ha infundido la dialéctica entre software privado y software libre.

Capítulo IV: Análisis de la valorización del conocimiento y la realización de ganancia en el caso de la AMESOL

Como hemos visto en los capítulos anteriores, la valorización del conocimiento en la realidad presente, tiene implicaciones que varían de acuerdo a las distintas formas en las que se realiza su restricción, donde al existir formas de restringir su acceso es posible realizar ganancia. Por un lado, en el caso del software privado, se hace a través de los instrumentos económico-legales de los derechos de autor. Por otro lado, el software libre como un "bien club", en donde la restricción del conocimiento se encuentra dada por la disposición y capacidad para realizar la apropiación del mismo.

La restricción económico- legal a la difusión del conocimiento formalizado, tiene la facultad de volver escaso el conocimiento de manera legítima y artificial, lo que también, propicia la posibilidad de que quien detenta los derechos sobre el conocimiento puede ejercer monopolio del mismo. Paralelamente a esta situación histórica, han aparecido nuevas formas de transformación del conocimiento, apoyadas en las nuevas tecnologías de la información, que permiten la reproducción y transmisión de conocimiento en tiempo real. El caso del software libre, nos presenta con un conocimiento formalizado que no puede ser apropiado en sí.[128] E incluso, con las características que le brinda ser un bien inmaterial o intangible, se le ha caracterizado como un "bien club" ya que tiene las características de ser conocimiento codificado.

En el presente capítulo, veremos que la práctica social del software libre, al encontrarse subsumida en una sociedad capitalista, se ha incorporado a esquemas de comercialización y realización de cuasi-rentas. Así, en un nivel abstracto, el conocimiento a través de su adaptación y transformación se configura como un elemento generador de valor en la medida que se ejerce como "bien club". Y además, esta práctica es capaz de realizar ganancias en el mercado.

[128] Sobre todo, cuando se trata de conocimiento que ha sido liberado bajo la GPL.

De esta manera, en el caso de la AMESOL, se hace posible ubicar cómo es que se ejerce aquella práctica. Atendiendo principalmente a dos situaciones: a) los modelos de realización de ganancia que se han descrito en el capítulo anterior y b) a las formas en las que se lleva a cabo la transformación del conocimiento, es decir, es decir su proceso de valorización.

En este apartado se presentan las prácticas de los sujetos que conforman AMESOL. Las cuales, se han podido colegir a partir de entrevistas a algunos de sus miembros por lo que no proclama validez estadística ni es posible aventurar proyecciones sobre el sector. En cambio, el análisis de las mismas no se realiza a nivel anecdótico, sino a través de su abstracción en categorías conceptuales. De esta suerte, dicho tipo de análisis ha permitido, que a partir de la construcción teórica que dirigió la primer aproximación al tema, se recogieran cuestiones no contempladas que han aportado de diversas formas el desarrollo conceptual de la tesis y no sólo de este último capítulo. [129]

Para realizar la exposición del actual capítulo, en primer lugar, se propone hacer una descripción más profunda de la asociación, y de los miembros que la componen. Así, se le puede advertir en su dimensión histórica, cuya emergencia y desarrollo atienden a una realidad cambiante desde un contexto específico. Después, en la segunda parte, se presenta una forma en la que se puede comprender la manera en que se realiza la valorización del conocimiento, desde sus implicaciones y circunstancias particulares que se han vislumbrado en esta realidad concreta. En tercer lugar, se atiende el negocio del software libre, a través de observar cómo se realiza esta práctica como estrategia de realización de ganancia. Por último, pasamos al plano que proporciona una visión más amplia, de lo que estas prácticas de valorización moldean, ya que poco a poco se han ido convirtiendo en prácticas generalizadas en la industria del software.

Si bien, este capítulo puede ser considerado descriptivo, tiene sentido en el cuerpo general de la tesis y ayudará a perfilar, en las Reflexiones Finales, el proceso que se estudia aquí como unidad de análisis. A saber: el doble

[129] Muchos de los profesionales de la industria del software, reconocerán partes en las que han contribuido u agradezco una vez más su confianza (véase: Introducción).

movimiento que presentan las empresas del software a través de sus prácticas en un marco histórico amplio.

1. La AMESOL desde, por y para sus miembros

Este primer apartado se encuentra dedicado a ofrecer un contexto en el cual se presenta un panorama de las bases históricas que han llevado a la formación de la AMESOL y de los propósitos que han tenido sus miembros para ella. Al considerar el ambiente institucional, las condiciones del mercado sin perder de vista el nivel de maduración de las tecnologías libres. Para ello, también caracterizaremos a sus miembros y lo que se han alcanzado de manera general como asociación.

Como ya se mencionó en el primer capítulo, la AMESOL surgió como una forma de agrupar y dar representación a la comunidad empresarial del software libre ante diversas instancias de gobierno, como lo es la Secretaría de Economía. Sin embargo, se requiere dar un panorama general de la industria del software que aporte el contexto histórico general, que conforma el entorno institucional del sector. Posteriormente veremos las motivaciones percibidas que llevaron a la formación de la AMESOL, para luego distinguir sus funciones y el modo en el que se encuentra conformada.

a. Contexto: el software en México desde los 80 y su repercusión en el presente

Para fines de los años 70, la industria mundial del software se encontraba ante la emergencia de dos empresas que marcarían el futuro del sector. Por un lado, el surgimiento la empresa de software Microsoft y por el otro, el surgimiento de Apple Computers, que inició la era de la microcomputadora para usuarios domésticos. No obstante, al observarlo desde el contexto mexicano, el panorama debe complejizarse.

Lo anterior se debe a la forma en que se desarrolló la industria en el periodo que va de principios de la década de los 80 al año 2000. Puesto que estuvo dada por la situación económica delineada por el modelo de sustitución

de importaciones y el paso al modelo neoliberal. Así como, el ritmo de las innovaciones en este ramo hicieron aún más difícil mantenerse al día.

Para ubicarnos históricamente, en primer lugar se menciona primeramente la situación de desarrollo en que se encontraban Microsoft y Apple. Esto se debe a que su influencia permitió la construcción de la industria del software mundial tal como se conoce hoy. Posteriormente se aborda la situación mexicana en particular, que definió la manera peculiar en que se insertó la industria en el país. En tercer lugar se hará un breve recuento de las distintas maneras en que empresas destacadas han influido en la industria del software mexicana. Por último, veremos las consecuencias que este devenir ha tenido en materia de políticas para el fomento de la industria del software.

En 1975 Microsoft empezó su modelo de comercialización privado, a base de licencias de uso, con su lenguaje de programación Altair Basic, para la computadora Altair 8800 de la compañía de electrónicos *Micro Instrumentation and Telemetry Systems* (MITS). Dicho modelo, fue el mismo que tuvo la oportunidad de afianzar con la aparición de la PC de IBM en 1981. Y para la cuál, comercializó el PC-DOS, como sistema operativo y con la posterior aparición de clones de la PC consolidó como un estándar dominante, bajo el nombre de MS-DOS.

Mientras tanto, desde 1977, Apple había introducido en el mercado mundial la Apple II, microcomputadora que marcó el éxito de esta compañía y para la cual, incluso Microsoft apoyó con programas.[130] A pesar de lo anterior, no fueron los programas de Microsoft los que marcaron el éxito del Apple II, sino el que escribió Software Arts Inc, y comercializado por Personal Software Inc. (ahora VisiCorp), el programa de hoja de cálculo llamado VisiCalc introducido al mercado en 1979 (Hormby, 2006; Licona et al., 1985), y aunque fue rápidamente clonado[131], transformó la microcomputadora de Apple en una herramienta de trabajo para las empresas[132] (INEGI, CFE, & MUTEC, 2003).

[130] Microsoft puso a la venta el SoftCard, dispositivo que le permitía al Apple II ejecutar software diseñado para otras computadoras (Licona, Angoa, Victoria, Bautista, & Burgos, 1985).

[131] Apareció SuperCalc (1980), MultiPlan (1982), Lotus 1-2-3 (1983), y el módulo de hoja de cálculo en AppleWorks (Wikipedia, 2010).

[132] "VisiCalc fue declarado el programa del año en 1982, por el «hit parade» que mantiene la revista Infoworld para productos de software. Su éxito es indiscutible, pues en 1983 VisiCorp

El surgimiento de la industria del software se encuentra entrelazada con el acceso de los programadores al hardware que hace funcionar y a la posibilidad de distribuir el software.[133] Por esto, es necesario observar que las condiciones para desarrollar software en México, se encontraron limitadas a la posibilidad de existencia del hardware en este mismo país. Con el auge petrolero entre 1979 y 1980 crecieron las importaciones en 175% y se gastaron 5 000 millones de pesos (Mochi, 2006), lo cual, en el contexto de la política económica imperante, se demandaba que de alguna manera se lograran sustituir dichas importaciones.

Por consiguiente, en 1979, durante el sexenio de José López Portillo (1976-1982), el Plan Nacional de Desarrollo Industrial contempla el fomento a la fabricación de sistemas de cómputo electrónico, sus accesorios y partes (INEGI et al., 2003; Mochi, 2006). De esta forma, se deriva el "Plan de Fomento Nacional para la Industria de Cómputo" (PFNIC), el cual, requería que el 51% de la inversión destinada a la producción de computadoras fuera de origen mexicano y el 49% restante, podría provenir del extranjero (Mochi, 2006; Barahona & I. Alonso, 2007). Aunque, la realidad de la crisis que aquejó a México en 1982 hizo evidente la dependencia tecnológica en que se encontraba el país. Al no poder producir los componentes esenciales que requerían las computadoras ensambladas en él, tal como lo exigía el PFNIC.

A pesar de las restricciones en las inversiones e importaciones, propias del modelo de sustitución de importaciones, Apple México inauguró su primera fábrica en 1984, con la producción de Apple II (INEGI et al., 2003). Empero, por problemas aduanales de importación y exportación ésta decidió irse del país en 1987, para regresar hasta 1992 (Mochi, 2006; Barahona & I. Alonso, 2007).

---

vendió 400,000 ejemplares de este programa, actualmente es el programa de mayor venta con una cifra récord de más de 700,000 ejemplares. En opinión de Steve Wozniak, cofundador de Apple, en una buena medida el éxito comercial de la Apple II se debió a la aparición de este programa. Mucha gente compró el Apple para poder usar VisiCalc" (Licona et al., 1985).

[133] Es evidente que la capacidad de comercialización del software, en esta época se debió a la aparición de sistemas de hardware periféricos que permitían cargar programas en la computadora. La aparición de lectores de cintas de cassette y discos magnéticos, fueron necesarios para que fuera posible el desarrollo de esta industria. De manera similar a la industria del fonograma, que a su vez se encuentra protegida por las mismas leyes de protección, los derechos de autor.

En esta época, la Apple II tenía notable presencia en México gracias a la utilización de Visicalc y se comenzó a explotar un área de oportunidad a través de cursos por parte de empresas para aquel programa. De esta forma, para competir con aquella empresa, IBM introdujo Lotus 1-2-3, integrados en su PC, a través de distribuidores como fue el caso de Execuplan (Mochi, 2006; López, 2010). En éste ámbito Siga Desarrollos, también tuvo éxito al personalizar hojas de cálculo, y logra importar las primeras PC a México. Del mismo modo, Digit apareció para distribuir software, instalarlo y brindar capacitación (Mochi, 2006).

Por otro lado, hubo notables desarrollos por parte de empresas mexicanas. En primera instancia, podemos mencionar el caso de ASPEL. Fundada en 1981 por el Ingeniero José Icaza, que comienza con paquetes programados en lenguaje *basic.* Y en la cual, su área comercial tiene un gran acierto al empezar la venta de programas a través de mayoristas entre 1986 y 1987 (Aspel, 2004; Mochi, 2006).

En segunda instancia, también se pueden mencionar los avances hechos por It Complements. Que empieza operaciones en 1980, y que en 1985 fue apoyada por el Consejo Nacional de Ciencia y Tecnología (CONACYT), para comenzar el desarrollo de la herramienta de programación Hotware. Dicha herramienta, fue presentada en 1988, y que resultó ser una aplicación de cuarta generación orientada a objetos, interoperable y multiplataforma.

Aquella herramienta, permitió a la empresa Kuazar Informática hacer paquetes administrativos que corrían en UNIX (Rodríguez, 2007; Corona, 2009) en forma cliente servidor. Sin importar así el sistema operativo del cliente. El desempeño de esta aplicación había sido tal, que posteriormente fue

> (...) también adoptada por CONACYT como la herramienta estándar para el desarrollo de sus aplicaciones operativas. Esta herramienta, resulta tan exitosa que la Comisión Federal de Electricidad (CFE) la elige a escala nacional para hacer desarrollos y posteriormente, el gobierno mexicano la selecciona para desarrollar el Sistema de Compromisos Presidenciales, el cual fue implantado en las 15 dependencias más importantes del sector público (Rodríguez, 2007).

Por otro lado, en 1986, se establece en México la empresa Microsoft, que erige acuerdos con fabricantes de computadoras para ofrecer equipos con sus

sistemas operativos preinstalados. Y cuyo costo de licencia era incluido como parte del precio final al consumidor. Así también, Microsoft convence a los mayoristas con respecto a la introducción de su sistema operativo con entorno gráfico Windows, al tiempo que introdujo la paquetería de procesador de textos (Word), hoja de cálculo (Excel) y presentaciones (Power Point). Para competir con el Word Perfect que dominaba en el MSDOS, y las otras hojas de cálculo como era el caso de Visicalc y Lotus.

Un año más tarde, en 1987, Oracle entra en el mercado mexicano, empresa dedicada a las bases de datos relacionales. Justo en este año, supera los 100 millones de dólares en ventas y se convierte en la empresa que maneja el software de bases de datos más grande del mundo con más de 4 500 usuarios en 55 países (The Gale Group, 2005). Donde en el caso de México resalta Pemex como su primer cliente (Mochi, 2006).

Como señala Mochi (2006), en el periodo de la década de los años 80, la oferta de software pasa de ser una "venta de productos" para convertirse en "venta de soluciones". Que llevó aparejada la posibilidad de tener servidores que realizaban el almacenaje de grandes bases de datos y realizaban tareas y proceso que eran solicitados, a través de redes[134], por computadoras menos potentes llamadas terminales o clientes. Como es el caso que permitió la herramienta Hotware o las bases de datos de Oracle.

También en esta misma década, fue evidente que existió la posibilidad de generar ganancia en la industria del software. A través de explotar lo que los programadores por lo general no hacían, la documentación y capacitación de los usuarios finales, como fue el caso de Digit.

Al final de la década que vio el surgimiento de la industria del software, perecería que la industria nacional mexicana se consolidaría, pero no fue así. Diez años después, en el año 2000, el presidente Ernesto Zedillo (1994-2000) convocó a varias empresas dedicadas al desarrollo de software, como son SofTeck, Microsof México, Praxis y Demis para crear el "Programa para el desarrollo de la Industria del Software" (PROSOFT), que se integra al proyecto

[134] En este momento empiezan a aparecer las redes de computo a nivel empresarial, separadas por unos cuantos metros o a través de países enteros.

e-México. El cual depende de la Secretaría de Economía, y como programa del gobierno de México, se compromete a aplicar políticas públicas necesarias para fomentar el desarrollo de software y apoyarlo a nivel de iniciativas estatales y municipales (Mochi, 2006).

Todo lo anterior, muestra que el sector del software es dinámico y se va adaptando conforme va cambiando la propia estructura y capacidades del software. Donde se pueden identificar olas que han movido la industria, como es el caso de hojas de cálculo, procesadores de texto, herramientas administrativas en red. Todas ellas, conforme avanza su desarrollo técnico han cambiando y moldeando el sector, la forma de producción, de distribución, de su comercialización y consumo.

El software libre presenta, entre muchas cosas, cambios en la que se realiza la producción del software y por lo tanto al ser subsumido como una práctica de la que se puede producir ganancia, ha generado cambios. Sin embargo, como esta breve reseña en el país muestra, las acciones de una compañía influyen en otras, presentando competencia, por lo que para mantenerse en el mercado es necesario adaptarse. ¿Qué pasa cuando las reglas de producción cambian radicalmente la forma en que se produce el software y la manera en que se hace posible la extracción de ganancia?

No obstante, antes de que sea posible pasar al análisis de la AMESOL, es importante detallar el contexto que hace posible su nacimiento de acuerdo a su entorno inmediato. Al final de esta reseña se hacen patente las condiciones que vieron el nacimiento del PROSOFT, esta política pública sintetiza gran parte de las acciones del gobierno mexicano ha tomado para impulsar el desarrollo de la industriaonstituido como un organismo intermedio que brindar acceso a este programa. del software en el país. Por lo que a continuación se describirá someramente esta política pública, puesto que la asociación que nos interesa, se ha constituido como un organismo intermedio que brindar acceso a este programa.

b. PROSOFT

Desde que el programa entró en vigor, en el año 2002, su objetivo principal era consolidar la competitividad de la industria nacional del software a largo plazo, y afianzarlo como líder en este sector para el año 2013 (Sampere & Buenrostro, 2008). Para esta fecha, se esperaba que se hubieran constituido 500 empresas competitivas a nivel internacional. Al emplear a 100 mil personas con exportaciones cercanas a los 5 000 millones de dólares (cercano al monto de la india). Lo anterior, sería igual a alcanzar 4.3% respecto al PIB, equivalente al promedio de la Organización para la Cooperación y el Desarrollo Económicos (OECD) (Mochi & Hualde, 2006, 2008; Mochi & Rivera, 2006).

Ahora bien, para lograr esto, el programa siguió las siguientes líneas de acción:

1. Promover las exportaciones y atraer inversiones.
2. Proporcionar educación y formación al personal en el desarrollo de software en cantidad y calidad convenientes.
3. Contar con un marco legal promotor de esa industria.
4. Desarrollar el mercado interno.
5. Fortalecer a la industrial local.
6. Alcanzar niveles internacionales en capacidad de procesos.
7. Promover el desarrollo de agrupamientos empresariales (Casalet, Buenrostro, & Becerril, 2008; Mochi & Rivera, 2006; Sampere & Buenrostro, 2008).

Para poder realizar estas tareas, se fundó en el año 2004, el “Fondo de Financiamiento del PROSOFT”, a partir del cual se realizarían financiaciones y patrocinio de proyectos tecnológicos. Donde el programa, tiene un esquema que requiere la existencia de Organismos Promotores. Los cuales, tienen el fin de distribuir los riesgos entre Entidades Federativas, los Organismos Promotores y el propio PROSOFT. Por lo tanto, es posible dividir las políticas concretas que el programa puede impulsar de la siguiente manera:

1. Capacitación.
2. Políticas de atracción y/o fomento de empresas locales.

3. Evaluación de la calidad del proceso del software.
4. Política de atracción de inversión extranjera.

Luego, sus políticas en cuanto a capacitación, se pueden dividir en: a) centradas en instituciones educativas, como lo son planes de estudio y equipamiento; b) orientadas a mejorar la vinculación entre academia y mercado de trabajo o empresas y, por último; c) apoyos a investigación y capacitación. En consecuencia, hasta el año 2006, había trabajado con 121 universidades, y además, habían apoyado a 1 200 desarrolladores de proyectos de capacitación (Mochi & Hualde, 2006).

Asimismo, en cuanto a sus políticas en atracción y/o fomento para empresas locales. Tenemos la aplicación de subsidios federales, que apoyan acciones para obtener como resultado: la generación de una masa crítica en las capacidades de los trabajadores. Lo que fomentaría, a su vez, proyectos productivos y oferta de servicios. Del mismo modo, también apoya con créditos del 50% del contrato o con crédito de cuenta por cobrar, al ocupar un fondo de "contra-garantías". Para el 2006 operaba con 25 gobiernos estatales y el Distrito Federal[135], donde se encontraban 10 conglomerados, formados por empresas pequeñas y medianas. Lo anterior, era con el fin de ofrecer soluciones y servicios de mayor alcance (Mochi & Hualde, 2006).

En cuanto al fomento a la evaluación de la calidad, se creó el MoPROSOFT y el EvalPROSOFT. El primero se diseñó como modelo de proceso para el desarrollo y mantenimiento de sistemas de software, que considera los modelos de evaluación Capability Maturity Model for Software (CMM), Capability Maturity Model Integration (CMMI) e International Organization for Standardization (ISO) (Mochi & Hualde, 2006, 2008; Sampere & Buenrostro, 2008). Mientras que el segundo, funciona como metodología de evaluación. Por consiguiente, en el 2004, se apoyaron 14 proyectos de iniciativas de calidad de los distintos modelos. Donde se apoyaron a más de 90 empresas que involucraban a 470 trabajadores. Igualmente, avaló a 18 empresas para ser evaluadas en modelos como el CMM, CMMI y

[135] Los estados que no se encontraban integrados son: Campeche, Estado de México, Guerrero, Hidalgo, Nayarit y San Luis.

MoPROSOFT. De la misma manera, asistió a 70 empresas para que pudiesen brindar capacitación y consultorías sobre modelos de procesos, como los antes mencionados (Mochi & Hualde, 2006).

Por último, la atracción de inversión extranjera, la realiza a través del apoyo a campañas publicitarias que dan a conocer las cualidades, capacidades y ventajas de México. Tal es el caso de la campaña "México IT: Always near your business", emprendida en Estados Unidos (Mochi & Hualde, 2006).

A todas estas actividades, se suma la tarea de coordinar institucionalmente acciones desempeñadas por diversos centros y organizaciones. Como es el caso del Instituto Latinoamericano de la Comunicación Educativa (ILCE), la Asociación Nacional de Instituciones de Educación en Informática, AC. (ANIEI) y la Cámara Nacional de la Industria Electrónica, de Telecomunicaciones y Tecnologías de la Información (CANIETI) (Casalet et al., 2008).

Después de todo lo anterior, resulta que el gobierno no ha sido el único que ha evaluado este programa. Lo ha hecho la Universidad Nacional Autónoma de México (UNAM) y la Universidad Autónoma Metropolitana (UAM), y sus recomendaciones han hecho posible que en el año 2008 se lanzara el "PROSOFT 2.0". Este nuevo diseño, tiene por objetivos articular y promover el desarrollo de la tecnología de la información. Donde se pueden incluir las actividades de *outsourcing* y la producción de software embebido, del mismo modo se busca articular los *call and contact centers* (Sampere & Buenrostro, 2008; Stezano, 2009).

La práctica del Programa para el Desarrollo de la Industria del Software, ha tenido impactos positivos y visibles en la industria del software. Existen estudios que ponen en evidencia dicha influencia en la evolución del sector, a nivel nacional (Mochi & Hualde, 2006), así como su influencia en programas regionales, como Aguascalientes (Casalet et al., 2008; Sampere & Buenrostro, 2008), Nuevo León (Casalet et al., 2008) y Jalísco (R. Oliver & González, 2008; Sampere & Buenrostro, 2008).

Por otro lado, el PROSOFT se puede complementar con otros programas

que han impulsado FUMEC, BANCOMEXT, PIAPYME y CONACYT para hacerlo más exitoso. Algunos de ellos tienen fondos mixtos, entre capital público y privado, que se orientan a la formación y desarrollo de capacidades tecnológicas en sectores y regiones (Casalet et al., 2008). A pesar de esto, cabe señalar la falta de prestamos y capital de riesgo para este sector (Casalet et al., 2008; Mochi & Hualde, 2006).

Otra cosa que se hace notar, son las limitaciones que tiene el mercado para desarrollarse de manera dinámica. Ya que las empresas no tienen la capacidad de demandar nuevos conocimientos científicos y tecnológicos, puesto que sus mejoras competitivas se basan en la compra de tecnología extranjera (Casalet et al., 2008). De la misma suerte, es un hecho que la circulación de conocimientos es escasa entre estas empresas. Por lo que falta impulso a la cooperación de diversas instancias para realizar investigación, desarrollo e innovación, donde ciertamente las PyMEs[136] resultan las más resegadas.

En este punto, justamente Mochi y Hualde (2006) destacan que en las evaluaciones de las políticas públicas faltan estudios que hagan evidentes dos aspectos de la industria. En primer lugar, el software subcontratado producido en el sector público mexicano. Y en segundo lugar, el papel que tiene el software libre, como una forma de hacer acceder a herramientas informáticas en sectores que no se consideran, como lo son las PyMEs.

En este punto, resulta interesante observar los inconvenientes que se presentan a nivel de la aplicación concreta de las políticas públicas. Sobre todo, a través de la teoría que hemos desarrollado en los capítulos anteriores. Donde podemos mencionar que el software libre, además de poder ser accedido sin

[136] Acrónimo de pequeña y mediana empresa. En México normalmente incluye a la micro empresa a menos que se indique lo contrario, por lo que se ha preferido al uso del acrónimo MiPyME. En este país se determina la categoría de acuerdo al número de empleados por el sector de la economía en el que se desempeña: Industria, comercio o servicios. Empero se considera microempresa a aquella que va de 0 a 10 empleados en cualquier caso. Por otro lado, se considera pequeña empresa a aquella que ocupa de 11 a 50 personas en el ramo de servicios e industria, y aquella que tiene de 11 a 30 en el sector de comercio. Por último, se considera mediana empresa a aquella que tiene de 21 a 100 empleados en los ramos de industria y servicios o bien, si se encuentra en la parte de comercio, a aquella que va de 31 a 100 trabajadores.

necesidad de pagar una licencia de uso -como conocimiento objetivado-, también puede ser apropiado como conocimiento codificado. Si esto es pensado así, resulta que todo aquel sujeto, sea empresario o individuo, se puede beneficiar de dos formas. La primera como apropiación de la cooperación pasada objetivada, pero también, de la segunda manera, como forma de acceso al conocimiento.

Ahora que hemos abordado las condiciones históricas que imperaban en el ámbito mexicano y el contexto de incentivos que otorgaban las políticas públicas, falta por abordar un tercer condicionante del surgimiento de la AMESOL. Esto es, el nivel de maduración que tenían las herramientas de software libre para aquel momento, así como el ambiente que se percibía y que requería ser cubierto dados todos estos factores preponderantes.

c. Contingencias para el surgimiento de la AMESOL

En el punto de inflexión que se da a finales de la década de los noventas. Que apuntaba a una industria mexicana que se consolidaría, y la aparición de políticas públicas destinadas específicamente hacia este sector. Ocurre que también se consolidaron herramientas de software libre que se encontraban a disposición de quien pudiera apropiarlas tanto como objeto como conocimiento.

A principios de la década del 2000 aparecieron programas robustos. Los cuales, competían directamente con productos ya establecidos en la industria internacional del software. Tal es el caso del *kernel* Linux[137], el servidor web Apache[138], el manejador de bases de datos MySQL[139], y los lenguajes de programación PHP[140] y Perl[141] (LAMPP[142]). Así como, al hecho de que varias

[137] Establecido en 1991 y con una versión confiable para 1994 y la reorientación del diseño del *kernel* hacia un enfoque modular a mediados de 1996, con la versión 2.0.

[138] Proyecto fundado en 1994 y consolidado en 1999 a través de la conformación de la fundación Apache.

[139] Constituido en 1995 y contando ya con su versión 3.23 en enero de 2001.

[140] Acrónimo recursivo de *PHP Hypertext Pre-processor*, es un lenguaje de programación diseñado para la creación de páginas web dinámicas, interpretador del lado del servidor, utilizado a través de una linea de comandos y aplicaciones, creado en 1995. Se afirma como software libre tras el establecimiento del PHP group en 2001.

[141] Es un lenguaje de programación iniciado en 1987, cuyo referente principal, el "libro del dromedario", fue publicado en 1991. El desarrollo de Perl se estabilizó en su versión 5, que comenzó con la instauración de la lista de correo *perl5-porters* en 1995.

[142] Así se le llama a la combinación de programas más comúnmente usados para poner en

empresas internacionales -como SAP- y Nacionales -como Softek y Kuazar Informática-, brindaban ERP's basados en estas plataformas.

En ese momento, existían figuras individuales que destacaban en el software libre en México. Quienes a su vez, se encontraban muy ligadas a los ambientes universitarios donde el software libre empezó a fluir a través de sus servidores. Entre ellos, podemos mencionar a Miguel de Icaza y Federico Mena, quienes fundan el proyecto GNOME[143] en 1999; así como personas que se consolidarían como desarrolladores del proyecto Debian[144], tal es el caso de Gunnar Wolf o Rodrigo Gallardo; personalidades destacadas en herramientas de programación específicas como Gerardo Horvilleur en Java[145] o bien; Roberto Andrade en PostrgreSQL[146]; así como técnicos con conocimientos especializados en software libre, como es el caso de Fernando Romo y Sandino Araico.

Sin embargo, a principios del año 2000 las herramientas de software libre -como las que mencionamos mas arriba- eran los suficientemente robustas como para erigirse en alternativas viables a las que existían y se comercializaban bajo el modelo de licencias. Por lo que se comenzó a perfilar la posibilidad de explotar la instalación y puesta a punto de software libre en servidores como un negocio posible. El cual, era factible, más allá de la existencia de figuras consolidadas y los ambientes académicos.

Aquí, podemos distinguir un doble movimiento microscópico, ya que empresas que se habían consolidado en mercado de software propietario, empiezan a introducir el software libre dentro de sus repertorios de aplicaciones y plataformas. Que les permiten brindar servicios confiables a sus clientes, al

funcionamiento un servidor corriendo con software libre.

143 *GNU Network Object Model Environment,* se ha consolidado como un entorno de escritorio orientado a proveer ambientes de desarrollo para software, así como software que pueda manejar el lanzamiento de aplicaciones, manejo de archivos, ventanas y administración de tareas, para brindar una interfaz de usuario atractiva e intuitiva.

144 Proyecto de distribución GNU/Linux, caracterizado por no contar con una empresa patrocinadora y ser mantenido por una comunidad fuertemente comprometida con los principios del FLOSS.

145 Java fue creado en 1995 por Sun Microsystems, sin embargo, el código fuente de su base, no comenzó a ser liberado sino hasta el año 2006, terminando el proceso en el año 2007, a excepción de una pequeña parte de la cual Sun no tiene derechos de autor (Martens, 2007).

146 También llamada simplemente Postgres, es software libre para manejar bases de datos relacionales.

migrar de UNIX de manera transparente a Linux, aprovechando el *expertice* del que ya disponían. Por otro lado, personas que se dedicaban a la administración de sistemas, adaptación y desarrollo de software comienzan a ver las ventajas de conformar micro y pequeñas empresas, para poder ser contratadas por el sector público y/o privado para brindar servicios al software.[147]

De esta forma, las empresas que se encuentran comienzan a dar soporte a plataformas y aplicaciones, como Linux o Apache. Pero de igual forma requieren también romper con la imagen que se tenía de los "*Linuxeros*", que en el ambiente en TI es de ser personas informales[148]. Como nos comenta uno de los miembros de AMESOL:

> (...) su informalidad, todos se visten de manera informal, y son informales. No llegan a tiempo, no llegan con una hojita de mira esto es lo que te voy a hacer, fírmale que estás de acuerdo, cuándo termine, recavo mi hoja de soporte y de servicio, me firmas de a qué hora terminé, que fue lo que te hice, no se les da eso. Son los *hippies* de la informática (Entrevista 7 a miembro de AMESOL, Marzo 2010).

De esta manera, nace de la propuesta de empresas que se dedican a brindar servicios, adaptación y desarrollos para y en software libre de formar la AMESOL. Y que además, se avocan a darle formalidad al desarrollo y servicio en software libre. Para ello, se empiezan a concentrar en vender soluciones y no misterios[149]:

> (...) los linuxeros, tienen también un problema, pues que se sienten gurús todos. En primer lugar, a mi me gusta vender soluciones, no problemas, entonces si yo puedo comprar un Red Hat empaquetado y que la gente pague 200-300 dólares por su suscripción, que creo que así le llama Red Hat, lo prefiero 200 veces, porque sé que si necesitan restaurar el servidor, ahí tienen su media, tienen la documentación, hablan

[147] Ya que de otra forma, se tendría que hacer a través de recibos de honorarios, los cuales dificultaban su contratación y pagos. Como nos comenta su experiencia un miembro de amesol: "Yo mucho tiempo estuve trabando como consultoría independiente y de pronto pues me invitan a trabajar en un proyecto, un proyecto grande para PEMEX. Y ahí la necesidad fue que necesitábamos facturas, no nos aceptaban nuestros recibos de honorarios, entonces ahí surge la necesidad de crear una empresa, en la que nos asociemos" (Entrevista 3 a miembro de AMESOL: Febrero 2010).

[148] Donde la cultura *hacker* es, "en esencia, una cultura de convergencia entre los humanos y sus máquinas en un proceso de interacción sin trabas. Es una cultura de creatividad tecnológica basada en la libertad, la cooperación, la reciprocidad y la informalidad "(Castells, 2001, pág. 66).

[149] Tal como comentó un experto en Linux en una entrevista: "A pesar de ser software libre, hacemos desarrollos particulares, nosotros decimos: regalamos los programas, compartimos el conocimiento, pero el misterio lo vendemos. Eso es nuestro negocio" (Romo, 2003).

> por soporte técnico a Red Hat, tienen soporte durante 6-8 años sin bronca, etc. Pero normalmente a los técnicos de UNIX eso es lo que no les gusta, les gusta sacar su versión de Debian [distribución de GNU/Linux] que les hacen tres, cuatro cosas así, nunca te dejan la media, ellos son los únicos que saben cómo lo instalaron, porque no documentan nada, y cuando se trata de replicar… (…) (Entrevista 7 a miembro de AMESOL: Marzo 2010).

Por esta problemática es que se decide crear la Asociación. La cual, permitiera a las PyMEs acceder al software libre. Que, como ya se ha señalado, tiene grandes ventajas como es: a) estar exento del pago de licenciamiento por uso y b) calidad técnica a través de brindar la experiencia de certidumbre en el proceso de soporte y formalidad en el trato. Así, empresas como Open Intelligence, dirigida por José Luis Chiquete Valdivieso, Tokonhu, fundada por J. Eduardo Moreno, Lingo Systems de Ruben Marrero y Ruth Menchaca Romo empresaria del software libre desde la década del 90, entre otros, crean en Marzo de 2003 la Asociación Mexicana Empresarial del Software Libre.

Como veremos más adelante con más detalle, la asociación creció en tamaño en los años siguientes. Este crecimiento, se debió a la facultad que como asociación tiene, para tramitar diversos fondos de fomento para el desarrollo de empresas y proyectos innovadores. Dicho de otra forma, la asociación se configuró como un organismo intermedio y/o promotor, que garantiza ante las instancias gubernamentales que se logrará generación de empleo. O bien, que habrá investigación, innovación o difusión de conocimientos, cuando se otorgan incentivos a sus miembros.

d. Constitución de un organismo intermedio

La AMESOL se ha constituido como un organismo que ha permitido a sus afiliados, acceder a recursos federales que en general, tienen el objetivo de fomentar el desarrollo del sector tecnológico. Entre los que se pueden destacar fundamentalmente seis programas.

En primer lugar, se encuentra el ya antes mencionado PROSOFT, ofrecido por la Secretaría de Economía. Para impulsar a la industria de software y extender el mercado de tecnologías de información. Por lo que la AMESOL se

ha constituido como un organismo promotor de este programa.

Así, y en segundo lugar, se ha consolidado como organismo intermedio de Fondo PyME, instaurado también por la Secretaría de Economía para brindar apoyo económico a las micro, pequeñas y medianas empresas. Para lograr su desarrollo y fortalecimiento en varios ámbitos, que van desde capital semilla, transferencias de conocimientos y tecnología; hasta apoyos para exportación.

En tercer lugar, se encuentra el "Fondo de Innovación Tecnológica", ofrecido conjuntamente entre la Secretaría de Economía y el CONACYT. Para apoyar proyectos de innovación tecnológica orientados a fortalecer la competitividad de las empresas mexicanas.

Por último, se encuentra el programa “Estímulos a la Investigación, Desarrollo Tecnológico e Innovación”. Que se compone de tres subprogramas: INNOVAPYME, INNOVATEC y PROINNOVA.

Para tramitar alguno de estos programas, se necesita pues, de una asociación que realice las gestiones adecuadas ante las instancias federales correspondientes. De esta forma, como incentivo para que se las empresas se volviesen parte de la AMESOL, se ofrecía cobrar un porcentaje menor al que se cobraba normalmente a los no miembros por concepto de consultoría y asesoría para poder realizar dichos trámites. Donde la cuantificación del costo por papeleo se realiza sobre el monto recibido del recurso federal.

Ahora bien, esto último tuvo gran influencia entre las empresas y marco el crecimiento de la asociación y su composición. A continuación pasamos a considerar, la composición que alcanzó esta asociación como organismo intermedio, ya que permitirá observar si se trata o no de empresas intensivas en conocimiento y la manera en que puede decirse si se apropian del software como conocimiento objetivado o como conocimiento codificado.

e. Los miembros de AMESOL y ¿software libre?

Más que hacer un listado de las empresas que conforman la AMESOL[150], conviene más bien, hacer referencias a qué tipo de empresas pueden llegar a formar parte de la misma. Lo cual, nos aporta más con respecto a la forma en que el conocimiento puede o no llegar a ser difundido y en la medida en que puede hablarse de empresas intensivas en conocimientos.

Como hemos visto, la AMESOL se constituyó como un organismo intermedio de programas impulsados por instancias de gobierno federales, como es la Secretaría de Economía y el CONACYT. Esto tuvo gran influencia en aquellos que se afiliaron a la misma.

> (...) muchos de los socios actuales, de AMESOL, se afiliaron por los fondos federales (…) yo me atrevería a decir que la mitad de los socios, llegaron aquí por eso, que además, nada tienen que ver con el mundo de open source, nada, nada, nada, nada, nada. Hay gente que no conoce OpenOffice, olvídate todo lo demás, ¡OpenOffice! No que no lo hayan usado, en su vida lo habían escuchado, «¿qué? Op... ¿qué?» así, esos son los socios que tiene AMESOL (Entrevista 1 a miembro de AMESOL: Diciembre 2009).

El cómo llegó a ser una asociación de software libre donde existen empresas que no integran ningún componente de software libre como parte de sus estrategias de negocios ni brindan ningún servicio relacionado con el mismo no es de nuestra incumbencia en este momento. No obstante, esta situación es digna de ser mencionada, ya que permite realizar una reconstrucción de la realidad de la AMESOL. De esta manera, es posible colegir el cómo hacen negocio los miembros que si efectúan prácticas relacionadas con el software libre, a través de la valorización del conocimiento codificado en el software libre.

De esta forma, podemos afirmar que hasta diciembre de 2009, había 26 miembros registrados. Que aportaban con sus respectivas anualidades a la asociación y entre las cuales existen empresas que si tienen actividades relacionadas al software libre. Aquí cabe la pregunta ¿qué es realizar prácticas de software libre en un marco empresarial?

Anteriormente hemos distinguido las motivaciones que puede tener una

[150] Lo cual se puede hacer desde su página WEB: http://www.amesol.org.mx

empresa para integrar el software libre a sus estrategias de negocios, empero ¿cabe hacer la diferencia entre usuarios y productores de software libre? Para responder a este par de preguntas, primero veremos cuáles son las prácticas que realizan los empresarios adheridos a AMESOL y posteriormente problematizaremos las contradicciones que puedan surgir de dicha *praxis*.

De AMESOL se pueden distinguir, dos tipos de socios que trabajan con software libre, el primer grupo lo conforma inmensas transnacionales que son IBM, Red Hat y Novell. Que hemos descrito someramente en el capítulo anterior. En el segundo grupo, se encuentra conformado por PyMEs, donde el resto de los que trabajan con software libre en AMESOL se encuentran en esta segunda categoría.

Así, podemos resumir que los socios grandes de AMESOL, se dedican, en el caso de IBM, a habilitar versiones de software propietario que corre en plataformas libres o bien, versiones extendidas de software libre que convive con software propietario e incluso a suscripciones a soporte de servidores Linux. Y de Red Hat y Novell, fundamentalmente se dedican a ofrecer suscripciones a soporte técnico y a las actualizaciones de sus distribuciones Linux empresariales, hechas por ellos disponibles en sus repositorios sólo a aquellos que se encuentran inscritos. O bien, brindan accesos a sus herramientas de software propietario, que corren en plataformas de código de fuente abierta, principalmente GNU/Linux.

Por otro lado, en las PyMEs, podemos encontrar sobre todo consultoría y desarrollo de sistemas a la medida. Lo cual, se traduce en adaptación de conocimiento de tecnologías ya desarrolladas y maduras a las necesidades del cliente, instalación, puesta a punto, administración y mantenimiento de sistemas.

> (…) es negocio instalarlo, es negocio ponerlo a punto, instalar el paquete que quieres correr, darle soporte técnico, ese es el negocio(...) (Entrevista 7 a miembro de AMESOL: Marzo 2010).

Sin embargo, también podemos encontrar que se utiliza como estrategia dentro de las empresas, es decir, como una herramienta de trabajo.

> Es nuestra herramienta de trabajo, pero al mismo tiempo es nuestra herramienta de

> venta y de todo, por ejemplo: dentro de la empresa no usamos la suite de oficina de Microsoft, usamos obviamente OpenOffice, todo el mundo lo usa todos los días, usamos en general Linux, aunque no todo el mundo tiene Linux en sus equipos, en los servidores que tenemos sí tenemos Linux y en los manejadores de las bases de datos que necesitamos también para las aplicaciones que tenemos, trabajamos con MySQL o Postgres[151], que son abiertos (Entrevista 3 a miembro de AMESOL: Febrero 2010).

Esto se debe en gran medida por el costo de las licencias que tiene para una PyME trabajar en el desarrollo de programas tipo ERP, si además debe pagar licencias de uso por los módulos de sistemas de gestión de bases de datos.

> (…) cuando una empresa, una PyME, te solicita una solución entonces es literalmente imposible que pueda comprar un producto de ese tamaño, por los costos. Entonces, eso de alguna manera te lleva a empezar a voltear hacia el software libre y buscar soluciones. Y una de dos, integras lo que ya existe, desarrollas algo nuevo o desarrollas un pedazo que te falte de algo que ya existe (Entrevista 3 a miembro de AMESOL: Febrero 2010).

Otra forma de realización de ganancia que se presenta en ambos tipos de empresas es la capacitación. A esto, las empresas grandes le llaman "certificación", que no es otra cosas que cursos para instalar, administrar y mantener software corriendo a nivel máximo de eficiencia. A continuación pasamos a conceptualizar más profundamente lo que estas prácticas significan en nuestro marco de análisis, la valorización del conocimiento en software y la realización de ganancia en el marco de la sociedad capitalista presente.

2. La valorización del conocimiento como "bien club" permite la realización de ganancia

En este apartado se caracteriza la forma en que se realiza la transformación del conocimiento, es decir, su valorización. Lo que no significa precisamente sólo su concreción en software, sino en un contexto que le brinda sentido. Dicho de otra manera, la producción de valor no sólo tiene que ver con el producir o

[151] MySQL y Postgres, son módulos de sistemas de gestión de bases de datos, que han sido liberado como software libre. El primero con GPL (también disponible bajo licencia comercial si lo desea el usuario, lo que se llama licenciamiento dual) y el segundo originalmente liberado bajo la MIT-style licence, que califica como software libre y ahora bajo la licencia PostgresSQL, aporovada por la Open Source Initiative.

desarrollar el software, sino en producir un software que cubra una necesidad dada bajo circunstancia particulares.

En este caso, el contexto tiene que ver con los recursos a los que tiene acceso el programador. Entre ellos, la basta biblioteca de programas que brindan las comunidades del software libre, a los que puede acceder a través de medios de distribución como Internet. Pero que, sólo puede aprovechar por los conocimientos que ha adquirido, ya sea de manera formal o informal, y si es capaz de ejercerlos de manera tácita.

Por otro lado, este "consumo" de conocimientos, tiene el objetivo de satisfacer las demandas del cliente, quien finalmente lo "consume" como valor de uso para constituirse en usuario. El silogismo que acabamos de describir, se refiere a la valorización del conocimiento codificado. Y sólo se puede consumar si se usa el conocimiento aprehendido, tácito, para poder transformarlo en software que cubra alguna necesidad específica. Esto sucede cuando se combina el conocimiento codificado (software libre) con el conocimiento tácito del desarrollador, para crear software que tenga un valor de uso. Lograr esto, requiere la capacidad de hacer esta síntesis, lo que excluye a de todo aquel que no reúna ambos tipos de conocimiento, y por ello se le ha llamado al software libre un "bien club".

Ejercer dicho "bien club" como un bien para el mercado, requiere la síntesis entre información y contexto. Así como la capacidad de realizar el trabajo de transformación de conocimientos explícitos, formalmente codificados, pero abstractos, en conocimientos formales para un contexto concreto y realizar una tarea específica. De este modo, podemos decir que se trata de un bien inmaterial o intangible que permite realizar una tarea concreta.

> (…) es más fácil que puedas vender esas adaptaciones a tus otros clientes, porque es fuente abierta, no le pertenecen al cliente, el cliente paga por que se lo mejores, pero las mejoras ya no le pertenecen al cliente, y ya no te pertenecen a ti, (…) (Entrevista 6 a miembro de AMESOL: Marzo 2010).

Esta tarea específica, que es capaz de realizar el software es lo que aparece como atractivo a su consumidor. Es aquello por la que vale la pena pagar, o más bien, invertir para que el software en "bruto" sea "refinado". A

través de la formalización de conocimientos abstractos, para transformarlo en un nuevo valor de uso, al adaptarlo a sus necesidades. Con la certeza de que puede llegar a beneficiarse de mejoras posteriores que otros le hagan al software que ha alterado.[152]

Dado lo anterior, en el subapartado siguiente, se caracterizan las implicaciones de apropiar conocimientos codificados y no sólo de apropiarse software como un "saber hacer" objetivado, como lo es comprar una licencia de uso de software producido en el extranjero. En el segundo, se detalla la forma en que se realiza esta valorización en la AMESOL. La cual, puede significarse en varios rubros, ya definidos más arriba, pero que en este momento conviene singularizarlos más puntualmente. Por último, se toman las implicaciones mutuas que tiene esta práctica y la industria de software constituida en torno a licencias.

a. Implicaciones de la apropiación, modificación y uso de conocimientos

En el marco de la sociedad capitalista actual, existe una industria del software construida en torno a licencias de uso que convierten el conocimiento en un bien escaso de forma artificial. Ahí, el software libre tiene sentido para las empresas grandes pues permite competir por nichos de mercado ya ocupados por aplicaciones dominantes, y por añadidura, con las empresas que los han desarrollado.

Utilizar el software libre como estrategia, exenta a las empresas, de las reglas del mercado de las estrategias impulsadas desde las restricciones económico legales del conocimiento. Así, es posible romper los candados del *lock-in* tecnológico e incluso por estrategias como el EEE, pero ¿cómo se pueden beneficiar las PyMEs de estas circunstancias para valorizar el software y crear ganancia?

[152] Si el código es liberado en términos de la GPL, ya que si se encuentra bajo la licencia BSD el código podría ser cerrado por aquel que haga modificaciones posteriores.

i. No importa el tamaño de la empresa, sino cómo se usa el software libre

Por más grande que sea cualquier empresa de software, no puede cubrir todas las exigencias que le pueden presentar los usuarios con altos conocimientos técnicos. Quienes funcionan como "miles de ojos", que se topan con problemas que no pueden resolver si se trata de software privativo.

> (…) en el [año] 98-99 (…) Netscape saca una suite de productos y con esos productos desarrollamos la herramienta para toda la gestión documental. Y precisamente con ese proyecto, nos topamos con un gran problema, a pesar de que se había comprado las licencias y demás, encontramos muchos *bugs* en los productos, *bugs* que no podíamos corregir porque no teníamos el código fuente(…) entonces pedíamos ayuda al proveedor: «Encontramos esto, esto, eso, esto, no funciona esto, necesitamos que...» y nunca nos dieron solución, entonces nosotros nos metimos en serios problemas con (…) el cliente. (…) [T]uvimos que estar dándole vuelta al asunto y parcheando las cosas por fuera, y de ese proyecto en adelante ¡jamás! Decidimos que: ¡jamás! volveríamos a usar un software del que no tuviéramos código fuente, porque ¡no!, te amarra de las manos, te vuelves totalmente dependiente de ellos (Entrevista 3 a miembro de AMESOL: Febrero 2010).

En este sentido, el software libre, brinda mayor control sobre las herramientas que se utilizan. En este punto, es posible advertir que se trata de una herramienta flexible en un sentido doble, por un lado para la empresa que lo desarrolla, y por otro, para la empresa que lo usa. Aunque ambos sentidos pueden coincidir en la misma empresa, conceptualmente esta distinción nos permite hablar de dos momentos, el primero en el que se crea su valor y el segundo en el que se realiza el bien como valor de uso, cuando se "consume". Y en ambos momentos, el software libre se presenta como altamente flexible, tanto como herramienta de producción como producto con diversas posibilidades de ser consumido.

> (...) la principal ventaja para una empresa que no la hemos sabido vender es esa, la libertad que te doy de que hagas con esta herramienta lo que te plazca, porque es software libre (Entrevista 1 a miembro de AMESOL: Diciembre 2009).

Y al mismo tiempo, es la libertad que los empresarios de software libre para hacer lo que ellos quieran. En contraste, en la industria del software

privativo, cuando se vende una aplicación se espera que el software genere *lock-in* al cliente y este regrese a comprar la nueva versión del software. Sin embargo, dado que se trata de conocimiento codificado, cualquier empresa o programador, puede tener acceso al código si considera que necesita alguna modificación.

> (…) la empresa más eficiente del mundo, el día de mañana a la mejor no lo es, la principal diferencia que tu tienes con el software open source, para un corporativo o para un gobierno, es que no estás cediendo a tu *roadmap*. Cuando tú le cedes a tu *roadmap* a una empresa de software cerrado, estás perdiendo mucha movilidad, ¿qué pasa si el día de mañana cambia la ley? (Entrevista 1 a miembro de AMESOL: Diciembre 2009).

El software se valoriza como "bien club" desde este punto de vista, en el momento en el que se consume como valor de uso para transformar conocimiento tácito en conocimiento codificado. Es decir, cuando se usa para producir un software con valor de uso. Donde en primer lugar, se requiere tener conocimiento tácito que habilite dicha transformación. Y en segundo lugar, se requiere también conocimiento que le permita solucionar contingencias.

No se trata de vender las licencias de uso, sino se trata de que exista aquel que tiene conocimiento para brindar servicios de instalación, puesta a punto, operación y mantenimiento de manera consistente.

> (…) entonces va a ser: "¿quien sabe manejar esta cosa?" No "¿quien me lo vende?" [sino] "¿quien me da el soporte?" entonces si va a ser el mejor estudiado, esa es la ventaja para el que se dedique al software libre (Entrevista 6 a miembro de AMESOL: Marzo 2010).

Este conocimiento no puede ser hecho escaso artificialmente, es más, impulsa la abundancia de su propio conocimiento. Tal como veremos en el próximo apartado, la comunidad de conocimiento en el caso del software libre tiende a divulgar su conocimiento. Esta abundancia de conocimiento, incrementa la flexibilidad de las empresas que apelan a él, ya que no sólo se trata del conocimiento que poseen, sino de las prácticas de conocimiento que les han permitido apropiarlo. De esta forma, es posible solucionar contingencias, puesto que si se encuentran con un problema desconocido entonces "hay que investigar".

ii. La flexibilidad y confiabilidad es cuestión de habilidad y tiempo

A través de los foros y listas correo electrónico de los diversos proyectos de software libre, es posible acceder a comunidades de conocimiento *sui generis*. Mismas que no son susceptibles de apropiación privada, ya que se trata de comunidades que alientan la construcción de la capacidad de hacer uso efectivo del software en cuestión, es decir, del conocimiento codificado.

> (…) a veces tienes un problema, te metes a los foros y la comunidad... encuentras una cantidad increíble de información (...) te topas con algún problema, te metes y ahí está la solución. Y es padre, porque finalmente encuentras algo. O tú haces algo y así como te encontraste algo que alguien hizo, pues lo publicas porque seguramente a alguien le va a servir (Entrevista 3 a miembro de AMESOL: Febrero 2010).

Así, es posible advertir, que en el caso del software libre como "bien club", es posible acceder al conocimiento si se tiene tiempo suficiente y habilidad. Esto es conocimiento tácito, que permite construir la capacidad específica de resolver una tarea concreta y con ello volverse miembro del club. Lo cual, se facilita si se realiza esta tarea sobre aplicaciones ampliamente usadas, ya que el número de personas en los foros tiende a ser mayor, con lo que se incrementa la posibilidad de encontrar una solución, o bien, que cuando se pregunta en el foro, alguien ya sepa la respuesta.[153]

En consecuencia, el propio uso generalizado de software libre lo convierte también en una herramienta robusta, a través de las aportaciones de otros desarrolladores y usuarios.[154] Las primeras a nivel de la valorización del conocimiento en su transformación en software y la segunda a través de la configuración de su valor de uso, al formalizar el conocimiento sobre su uso específico en un contexto dado. De este modo, el acceso a las comunidades impulsa el conocimiento de las herramientas que crecen en popularidad.

> Struts no era un framework muy bueno para desarrollo, pero no mucha gente lo usaba,

[153] Por ejemplo, de la distribución Ubuntu GNU/Linux, una de las más usadas en el mundo (DistroWatch, 2010), su *forum* oficial (en ingés) al 27 de Julio de 2010, contaba con un total de 1 415 048 hilos, con 9 074 943 posts. Y donde existían 1 118 648 millones de miembros, y hasta entonces, el momento en que más usuarios registrados en línea ha habido conectados al mismo tiempo fue el 10 de diciembre de 2009 con 37 719 (Canonical, 2010). Sin contar los foros regionales, tanto oficiales como no oficiales, que tratan temas relacionados con esta distribución.

[154] Razón por la cuál, diversos estudios aseguran de que el software libre se beneficia de los *free riders* (Lancanshire, 2001; M. A. Smith & Kollock, 1999).

> entonces, ¿qué fue lo que pasó? Que la gente que hizo Struts ha ido mejorándolo, mejorándolo, mejorándolo y actualmente es un framework muy robusto para desarrollo, la misma comunidad te va empujando (Entrevista 3 a miembro de AMESOL: Febrero 2010).

De esta forma, es posible entender que el conocimiento codificado como software libre, tiene sentido para las micro empresa. Ya que ese conocimiento les brinda flexibilidad y redefine su capacidad de hacer con el software.

Este tipo de software ofrece la posibilidad de elección. Es una alternativa al en torno a software privativo, donde existe *lock-in*. Puesto que ofrece un ambiente en el cual: el haber aprendido a aprender permite solucionar los inconvenientes que se presenten. Tanto a nivel de ofrecer valor de uso al cliente, como de valorizar conocimiento para usufructo de la propia empresa:

> (…) yo tengo mucho contacto con el cliente, y hay muchos mitos: «Yo no trabajo el software libre porque ¿quién me va a dar soporte técnico?» o «yo no trabajo con software libre porque no hay las aplicaciones que yo requiero». Y en estos casos, yo me he sentado con ellos y les he preguntado ¿qué aplicaciones tu requieres que no esté en software libre? Entonces desde gestores de contenido hasta soluciones para enviar correos electrónicos, soluciones financieras y te puedo decir que en el 85%-90% de los casos les he podido ofrecer una solución, y una solución de alto valor.(…) Nunca nos hemos enfrentado con un problema técnico que no hayamos podido solucionar, y que tengamos que utilizar una herramienta propietaria (…) (Entrevista 4 a miembro de AMESOL: Febrero 2010).

Para ejercer el software libre como un “bien club”, en necesario romper la barrera de la apropiación del conocimiento, lo que requiere la comprensión de la complejidad que requiere la alteración del código y el tiempo requerido para aprender el conocimiento (aunque socialmente disponible) y aplicarlo (desarrollar software con ese conocimiento aprendido). La barrera aquí, es el tiempo socialmente necesario para adquirir conocimiento tácito y transformarlo en conocimiento explícito. Y la apropiación de este conocimiento, brinda flexibilidad a una empresa ante circunstancias que cambian rápidamente,

Ahora bien, la posibilidad de transformar esto en ganancia requiere: la capacidad de proyectar la capacidad de transformación del conocimiento al público “consumidor”. Pues es éste, quien potencialmente puede requerir los servicios de las empresas del software.

> Generalmente son conocidos, les diste un curso en tal lugar y de ahí te conocen (…), es más internet, últimamente han llegado muchas cosas a través de internet, a través de la pagina(...), oye ¿cómo te enteraste? ah pues por internet (Entrevista a miembro de AMESOL: Febrero 2010).

La respuesta a este dilema es social, no se trata de aquel que puede restringir el acceso al conocimiento -convirtiéndolo en un bien escaso-, sino en aquel que es capaz de construir su lugar en la comunidad, no haciendo cosas, sino enseñando como se hacen, así se demuestra que tiene capacidad de transformar el conocimiento. Y si el problema supera el conocimiento disponible y el tiempo requerido para hacerse de ese conocimiento es prolongado, se preferirá pagar para que alguien más realice dicha tarea.

iii. En suma

Podemos mencionar que el software libre se traduce en conocimiento que es susceptible de ser apropiado dados los conocimientos suficientes y tiempo. Lo cual, brinda flexibilidad tanto a las empresas que lo utilizan como herramienta, como para aquellas empresas que invierten en sus adaptaciones, que satisfacen alguna necesidad particular.

De esta manera, cuando se trabaja con software libre, importa tener conocimientos o la habilidad para acceder a ellos. Así es posible ofrecer calidad y formalidad que ahorren o den certeza sobre el tiempo que consumirá el tener el producto o servicio deseado. Donde y aunque el conocimiento se encuentra socialmente disponible, no se puede invertir el tiempo necesario para transformarlo en valor de uso, es decir, conocimiento formalizado o codificado.

> (…) me han tocado personas que dicen «sabes que yo no voy a tomar tus cursos de 6000 pesos porque fulanito de tal me lo da en 2000». Entonces si lo quieres tomar con fulanito de tal adelante, estamos para apoyarte y hay personas que lo toman y a los tres meses regresan con nosotros porque el curso que se les impartió no fue de calidad o personas que: «Fulanito de tal me está dando el soporte a mi servidor Linux y el servidor de correos se me acaba de caer, y le hablé a este cuate y: [1] no me contesta o [2]; está de vacaciones en Acapulco y me dice que me puede atender hasta la próxima semana» (Entrevista 4 a miembro de AMESOL: Febrero 2010).

Para saber quien ofrece estas cualidades, se apela al reconocimiento de

la comunidad, la reputación, que permite determinar el "quien sabe usar esta cosa". Aquel que no sepa y tenga una necesidad, se acercará a estas comunidades de conocimientos, ya sea a través de Internet o bien a la red social de conocidos instruidos en el tema. Ahí habrá quien destaque por su prestigio, aquel que tendrá mas posibilidades de cubrirla. Sin duda, el precio es disuasivo, pero también lo es la calidad de trabajo desempeñado, que sólo puede cubrirse con los conocimientos que proporciona la habilidad de conseguirlos, pero mas allá de eso, optarán por aquel que les haya vendido una solución, no un misterio.

b. Taxonomía de la relación dialéctica entre valorización del software libre y concreción de la ganancia

En el apartado anterior, definimos términos en los que se presenta la valorización del software libre en la AMESOL, pero ahora definiremos cada una de ellas. Esta exposición se ha dividido en dos partes, la primer parte se refiere al conocimiento que es necesario en particular para valorizar el software libre, ya que éste permite modificarlo. La segunda versa sobre el conocimiento sobre el software libre en sí.

i. Posibilidades de valorización y realización de ganancia

A continuación se presenta una clasificación de prácticas de los empresarios del software libre donde el conocimiento es necesario para valorizarlo:

- La adaptación de software. Implica fundamentalmente dos cosas, en primer lugar cubrir una necesidad, evaluarla y si presenta una situación desconocida aprehender la manera en que se ha resuelto, para así hacerse de un conocimiento específico desde un contexto particular.

  (…) nosotros vemos qué tenemos para cubrir esa problemática y si no tenemos algo investigamos. Si no sabemos, échate un clavado en el tema, qué tecnologías abiertas maduras, con capacidad de ofrecer cubrir una necesidad de manera real, sólida y si lo tenemos se le ofrece al cliente (Entrevista 5 a miembro de AMESOL: Febrero 2010).

  Así, en segunda instancia, requiere capacidades que le permiten a los miembros de la empresa solventar problemas a partir del "saber hacer",

es decir, el haber aprendido a aprender. El cual, es un saber tácito que les permite apropiarse de conocimiento codificado, que es otro "saber a hacer", pero objetivado, formalizado en software que cubra las necesidades específicas del cliente. Y donde realizar este trabajo tiene un precio.

- Instalación. No basta con hacer la adaptación del software, es necesario que el hardware pueda acceder a ese software y sea capaz de correr el programa. Por lo que se requiere de alguien que haga esto posible y se hace ha través del pago por este servicio.
- Puesta a punto. Una vez instalado, es necesario que el software funcione de manera óptima en un hardware específico e interactuar con otros módulos de software. Esto significa, que se requiere hacer configuraciones específicas al software que se ha adaptado. De la misma manera, en algunos casos, incluso desarrollar software que permita el desempeño idóneo del mismo[155], lo que implica invertir esfuerzo y tiempo vital para realizarlo.
- Administración y mantenimiento. Los sistemas computacionales conectados a Internet, se encuentran expuestos a diversas contingencias, que no son deseadas por los usuarios que lo han puesto en funcionamiento. Ya sea por: a) ataques, hechos por usuarios malintencionados a través de explotar agujeros de seguridad, o bien; b) malware[156], sin embargo hay que destacar que el software libre es mucho menos propenso a sufrir inconvenientes debido a estos programas.[157]

[155] Como es el caso de *drivers, plugins* y/o *scripts*.

[156] *Malware*: es un término que refiere a todo tipo de software producido con la intención de realizar tareas no autorizadas y potencialmente perjudiciales para los intereses de aquel que lo ha puesto en funcionamiento (aunque muchas veces sin saberlo). Este tipo de software incluye entre sus tipos más conocidos: virus de computadora, gusanos, troyanos, *spyware* y *adware.*

[157] Esto se debe a los miles de ojos que pueden ver el código fuente (véase: Capítulo II), estos hacen factible detectar más rápidamente una flaqueza o que se está explotando alguna vulnerabilidad, esto permite que pueda ser reparada en el acto si se tienen los conocimientos y la solución puede ser compartida al resto de los usuarios a través de proponer actualizaciones o bien, alertando sobre la existencia de problemas en los foros de las comunidades del software. De esta forma, no se requiere que una compañía, responsable de algún programa antivirus detecte, proponga la actualización y el programa sea capaz de eliminar el software, o se deba esperar a que la empresa que hace el software arregle el

Pero también se pueden deber a: c) fallas del hardware, así como; d) a problemas con actualizaciones que hayan roto paquetes modificados por los propios usuarios o que hayan sido forzadas en plataformas obsoletas, etc. Para todas ellas y las que podrían surgir, se requiere de soporte técnico, encargado de restablecer el sistema, cuando deja de funcionar y dosificar sus recursos para que éste se mantenga en funcionamiento óptimo el mayor tiempo posible.[158] Aprehender esta ocupación, requiere conocimiento tácito, que ha sido adquirido a través del propio contexto y la experiencia pasada del administrador.

Bajo el modelo privativo de comercialización de software, se paga valor de cambio por el derecho de uso del mismo, no por el software en sí. De esta forma, en el mercado de licencias, en la mayoría de los casos se obtiene un producto estándar. En cambio, en el modelo empresarial del software libre es posible encontrar atractivo pagar por convertir: el saber hacer codificado en un valor de uso a la medida de las necesidades percibidas, para todas estas prácticas es necesario transformar conocimiento tácito en conocimiento formalizado.

Por otro lado, al hablar de conocimiento, y en particular del que circula como software libre, hay que notar la particularidad de que se trata de conocimiento al que sólo tienen acceso los que cuentan conocimientos suficientes para modificarlo. En consecuencia, para administrar un sistema no es necesario haberlo codificado. Por ejemplo: para administrar un servidor, no es necesario conocer los algoritmos que lo hacen funcionar, sino sólo los comandos que ejecuten las tareas deseadas.

---

problema, y publique la solución. Y aunque sistemas operativos como Linux no son inmunes a los virus, nunca se han encontrado virus que afecten a estos sistemas en la misma proporción que afectan a los sistemas de Microsoft (Granneman, 2003). En este tema, el riesgo se encuentra reducido por la utilización de repositorios con validación de la integridad de los datos (*checksum* o *hash value*) que permiten confiar en que el software que se descarga no ha sido alterado, así como la estructura de multiusuario que se ha heredado de UNIX, donde un virus que infecte la totalidad del sistema requiere que sea ejecutado por el usuario raíz (*root*), el cual se encuentra protegido por contraseña.

[158] Lo que implica que sólo se detenga para realizar el mantenimiento preventivo y necesario de manera controlada.

ii. El conocimiento sobre el software le brinda valor de uso

Para convertir el software libre en un bien útil, es necesario que sus usuarios sepan utilizarlo. En su desarrollo inicial, el propio software libre fue un instrumento creado por los usuarios que ya contaban con altos conocimientos técnicos. Sin embargo, al popularizarse este software se ha hecho necesario enseñar sus principios técnicos, y al hacerlo valoriza el software al hacerlo útil para el consumidor.

> Lo que yo les digo «en un servidor Windows él te configura todos tus permisos y tú aquí puedes configurar todos tus permisos». En cuestión de capacitación, la mayoría, un 50-60% de las personas, los primeros días de la capacitación tienen miedo, no saben que onda, les moviste todo el mundo, pero al final de la capacitación salen muy contentos porque ven que aprendieron y además de que son libres de hacer lo que ellos deseen con el sistema operativo (Entrevista 4 a miembro de AMESOL: Febrero 2010).

De la cita anterior es posible poner en relieve que, al menos en esta etapa del desarrollo del software libre, el conocimiento de sus características útiles requiere de conocimientos que no son asequibles de manera regular. O sea, que requieren de cierto esfuerzo que permite aprenderlo.

Más arriba se mencionó que no es necesario conocer sus principios técnicos para manejar software libre. Pero siempre ayuda conocer “algo” de programación si se quiere ejercer todo su potencial. Ya que cuando se aprende el cómo funciona es posible modificarlo.

En el esquema de software privado, el soporte se da por sentado y se espera que existan diversas garantías de su funcionamiento. Como es el caso de una instalación sin inconvenientes, números telefónicos disponibles las 24 horas del día y que la capacitación es responsabilidad del proveedor. Todos estos costos se cubren a partir de la venta de las licencias de uso. Al contrario de lo que pasa con las aplicaciones libres, donde la instalación, servicio y la capacitación son sólo responsabilidad del usuario final, es él quien tiene que pagar directamente por estos servicios.

No obstante lo anterior, es necesario remarcar que si existen entes que prestan dichos servicios y que estos forman parte de la AMESOL. Tanto desde la gran empresa como desde PyMEs, los primeros ofrecen suscripciones a

soporte técnico y certificación, mientras que los segundos ofrecen consultoría, soporte y capacitación que incluso puede llegar a estar certificada por los primeros. Dado que, el conocimiento de estos sistemas está ahí, disponible para cualquiera con aptitudes y tiempo suficiente. Lo que posiblemente cambie es la estandarización que ofrecen los primeros, similar a la producción en serie, contra soluciones a la medida que pueden ofrecer los segundos, que asemejan más a una producción artesanal. En seguida, observaremos cuales son las consecuencias de la capacidad de utilizar estos conocimientos en el marco de un mercado que se había conformado en torno al modelo de ganancia de licencias.

c. Circunstancias de una relación dialéctica de oposición para la valorización de conocimiento

Cuando se realizó la breve reseña histórica del software libre (véase: Capítulo II), se puso de manifiesto que tiene imbricado un proceso en el cual se estableció como dominante el modelo que restringe el conocimiento y lo transforma en un bien escaso de manera artificial (el software propietario). Esto conllevó a que se construyera el calificativo de libre, siendo que al principio, todo software lo era de *facto.* Del mismo modo, existen teóricos de este fenómeno que han puesto de relieve que: un proyecto de software libre tiene más posibilidades de ser exitoso si se tiene el sentimiento de que se encuentra haciendo algo “noble” o que existe oposición contra algo “malvado”, *verbi gratia*: Steven Weber (Weber, 2000).

Ahora existe la corriente del *open source*, que advierte sobre la necesidad de crear una imagen menos alejada de la empresa. La cual, se preocupa más por el desempeño técnico del software, que por discusiones ideológicas. Y que reconoce la necesidad de acercarse a trabajar “codo a codo”, con software privativo, para realizar el trabajo (véase: Capítulos II y III).

A continuación, se presenta la forma en que se percibe el uso de software libre frente al software privado. Al tiempo que se retome el cómo es que los miembros de la AMESOL lidian con el hecho de reconciliar la

convivencia del software libre con el software privativo. O sea, el hecho de que el software libre haya nacido como una forma de escapar a la apropiación del conocimiento legitimada por el orden económico-legal imperante (el capitalismo). Para luego, fuera integrado a este mismo orden económico pero con una resignificación de lo legal justificada por ser percibida como una mejor opción técnica.[159]

Así, el software libre como una opción que es capaz de ofrecer un mejor producto a los clientes y sin los inconvenientes de la existencia de las restricciones legales del conocimiento o deficiencias técnicas. Donde, se apunta a la empresa Microsoft como:

> (...) apoteosis de lo que está precisamente mal con el desarrollo del software, tanto socialmente (despiadadamente compitiendo a cualquier costo para maximizar la ganancia, en la misma medida en que se opone a promover verdadera innovación) y técnicamente (produciendo burdo, complicado y voluminoso software que genera *lock-in* a los clientes sin cubrir sus verdaderas necesidades como usuarios de computadoras) (Weber, 2000, pág. 20).

La referencia a Microsoft, destaca en el 100% de las entrevistas a los miembros de AMESOL. Si bien no contra esta empresa, sí contra la actitud estratégica que ha tenido para con el software libre. Aquí destaca el papel que tiene la piratería de sus programas en mantener una base de usuarios que, si fuera por la aplicación y cumplimiento de la ley cabalmente, estos no debieran tener acceso a este software en primer lugar. Por lo que se percibe que Microsoft tiene una postura que podemos calificar de hipócrita, al calificar a sus usuarios cautivos de piratas y al mismo tiempo de fomentarla (véase: El siguiente apartado).

> La gran empresa monopolio, de este lado, la principal diferencia desde el punto de vista de vender este producto... históricamente se le ha vendido al corporativo o al agente de

[159] Un aspecto del software libre, que se ha puesto de manifiesto, es que al producirse es propenso a tener un menor número de *bugs* que sus contrapartes comerciales así como su seguridad, tanto contra ataques, como contra *malware*: “Hay empresas que han hecho ya su migración completa a sistema operativo Linux, fundamentalmente te quitas de encima de los problemas de los virus, que a la fecha tenemos un cliente que se infestan sus servidores de Microsoft de virus y andan locos vacunando todas las máquinas que tienen en red y están terminando y ya el servidor les distribuyó otro virus, entonces lo sufren, lo sufren y siguen con lo mismo (Entrevista 3 a miembro de AMESOL: Febrero 2010).” La referencia a los virus, tiende a aparecer como evidencia de la eficacia con la que se produce el software libre y además, una buena razón para que los clientes se pasen a las opciones de fuente abierta.

> calle diciéndole: “es que te cuesta más barato”. Tú tienes dos sectores, el sector de casa, al que le dices: "es que te cuesta más barato que tu Office que vale 4-5 mil pesos y tu nuevo Windows en una cantidad parecida entre 8 y 10 mil pesos una cosa que te deje trabajar, no la versión mas baja de Windows (…). ¡Pero no es cierto! Eso es irreal, a ellos no les cuesta, a ellos les cuesta 30 pesos. Es triste, pero es la realidad, es decir, hay un mercado negro donde toda la gente que manda a hacer sus máquinas o que actualiza una nueva versión no compran en Home Depot, no compra en Wal Mart, mucho menos lo compran directamente en Microsoft. Ellos van a república del salvador o con su pirata de cabecera a que les provean algo por lo que están dispuestos a pagar, que son 30 pesos. Es decir, a ellos no les puedes vender por precio. Al corporativo tampoco, porque el corporativo tiene presupuesto para pagar cantidades mucho mayores y el precio no les importa (Entrevista 1 a miembro de AMESOL: Diciembre 2009).

Ante esta estrategia, que incluso ha sido reconocida por directivos de Microsoft (Mondok, 2007) el argumento de que el software libre es más barato en términos relativos para los consumidores menoristas y mayoristas, tiene sus bemoles. Puesto que tienen que sopesar el costo de los servicios que implica en una sociedad donde la piratería existe y por lo tanto, se debe añadir el cálculo que implica utilizar el software de manera ilícita.[160]

> (…) cuando una empresa, una PyME, te solicita una solución entonces es literalmente imposible que pueda comprar un producto de ese tamaño, por los costos, entonces eso de alguna manera te lleva a empezar a voltear hacia el software libre y buscar soluciones. (Entrevista 3 a miembro de AMESOL: Febrero 2010).

Decir lo anterior, se refiere a que no se puede aspirar a que una PyME opte por una licencia legítima del software. De esta manera, por un lado, el costo de las licencias (fundamento económico legal de la apropiación del conocimiento) es una barrera para acceder de forma legítima a la sociedad del conocimiento. Y por el otro, fomenta la apropiación del software como cosa, no como conocimiento codificado.

De esta forma, es posible advertir doble movimiento, en cuanto a la dinámica de la realización de la valorización y la realización de la ganancia a partir de software libre. Por un lado, el movimiento que ha llevado a la legitimación de la industria del software basada en licencias, es decir, la

[160] Para una discusión más amplia sobre las implicaciones económicas, políticas, jurídicas y sociales de la piratería véase: Cano (2006).

apropiación privada del conocimiento. Y por otro lado, en ese mismo marco de legalidad, se advierte al software libre, como la solución a realizar la cosificación del conocimiento, o dicho de otra manera: el software libre permite la apropiación del software a usuarios, como conocimiento codificado y convertirse en desarrollador. Cuyas capacidades estarán dadas hasta el nivel que le permiten sus propios conocimientos, habilidades y recursos económicos, tal como sucedía en el ambiente d el software antes de que fuera necesaria la distinción de “libre”.

Visto así, la piratería pudiera presentarse como una paradoja. En primer lugar como un motivo para producir el software libre. Ya que no se necesita realizar el pago de licencias de uso, se acaba con el estigma de violar la ley, que en este plano, se advierte como un motivo que tendría el usuario para cambiarse al software libre. En segundo lugar, aparece como la causa de que el usuario no migre al software libre, porque tiene acceso a la tecnología aunque sea pirata.

En efecto parece una paradoja, que de forma lógica parece irresoluble. Sin embargo, el software libre existe, hay usuarios que lo han adoptado, y además su uso continúa en crecimiento. Esto se debe a que la realidad tiene tiempo y los sujetos, que ejercen prácticas sociales en ella, tienen memoria del pasado y perspectiva de futuro.

> (…) yo creo que el mensaje es muy claro, open source, va a seguir creciendo. Microsoft en su forma de hacer negocio, cuando ya vea que en el sistema operativo y en la opción de office ya el mercado, no está dispuesto a pagar eso. Microsoft va a cambiar su forma de hacer negocios, (...) Microsoft tiene que irse a crear otro tipo de productos más especializados, mayor valor agregado, que la gente si esté dispuesta a pagar, porque ya el sistema operativo con todas las fallas del Windows Vista y la suite de *office.* Mira, la mayoría de la gente escribe sus hojitas, su Power Point, su Excel todo eso, (...) la vedad es que (...), la gente no es tonta, Microsoft va a cambiar su forma de hacer negocios. No porque ella quiera, sino porque el mundo exterior se lo va a imponer (Entrevista 2 a miembro de AMESOL: Febrero 2010).

Lo que se presenta aquí, es que la práctica del software libre para valorizarlo y realizar ganancias, encierra una doble subjetividad, como reconstrucción del pasado (memoria) y apropiación de futuro (utopía). Así, esta

práctica es un anudamiento entre el pasado, motivo de creación del distintivo "libre" y, por el otro lado, un futuro deseable, en donde se use un software que se percibe como técnicamente superior. Donde el verdadero plus del software libre, que se ha señalado es la flexibilidad que brinda éste, tanto para sus desarrolladores a la hora de usarlo como herramienta, como para las empresas que lo utilizan sin estar supeditadas a algún *lock-in* tecnológico que beneficie a otra empresa.

Así, la práctica del software libre, se yergue como posibilidad de incidencia en la realidad, una realidad que no se percibe como dada, sino como un dándose, en la cual se puede cuestionar desde la propia experiencia-conciencia.

A continuación, atenderemos lo que estas prácticas han cambiado en el plano de la visión general de la industria y la forma en que han influido en las prácticas de actores a nivel de las empresas más grandes del sector. Si bien, no se alcanza el ideal de quienes implementan el software libre, éstas si han tenido repercusiones notables en la configuración de la realidad actual.

3. La síntesis de dos modelos en el devenir del presente: Un doble movimiento accidentado

Es claro que hoy en día el software libre tiene una clara influencia en la industria del software en general, donde se aprecia un doble movimiento a nivel global. Ya que empresas de la talla de IBM o Novell, han cambiado hacia modelos que integran el software libre en sus modelos de negocios, también hemos visto que el software libre tiene sentido para la PyME, por multitud de razones, que van de la inexistencia de precios por licencias de uso, acceso legítimo a software de alta calidad, pero la más importante de todas es que permite la apropiación del software como conocimiento.

Este contexto, también ha cambiado la realidad de la industria del software en general, donde la principal compañía del software propietario, Microsoft han cambiado su comportamiento ante la fuente abierta, en cuestiones sutiles, pero que aparecen en la realidad concreta de sujetos que

integran la AMESOL.

> Porque qué fue lo primero que intentó hacer Microsoft, fue tratar de comprar empresas Linux[161] para tratar de quitar a la competencia, pero no puede comprar a todos. Incluso se está asociando con IBM en algunos proyectos y, siento yo, que Microsoft está doblando las manos, porque, la gran ventaja es que... yo no me explicaba... a mí me tocó la primera versión más o menos buena de Windows, las 3.1, que requerías 20 megas en disco y en memoria 64k y la Mac, con menos memoria corría más eficiente que Windows. Yo creo que fue un pacto no escrito entre Microsoft e Intel: «¿sabes qué? Yo voy a hacer, que las nuevas versiones de mi sistema operativo requieran de tu nuevo procesador y más memoria». Y a los fabricantes esto les parece la mejor idea del mundo, porque cada vez que salía una nueva versión de Windows requerías más máquina y te obligaban, casi casi a tirar tu máquina y a comprar una nueva y nos metieron en un rollo mercadotécnico (...) y eso fue muy claro hasta el último Windows Vista, ya que el mercado dijo: «¡no espérate!, ni es bueno y requiere muchos recursos» (...) yo creo que Microsoft hacía eso a propósito, hacer que sus máquinas requieran muchos más recursos que otros sistemas operativos, pero les había funcionado hasta le fecha, ahora el mercado dijo: «espérate tantito, ya hay otras opciones, ya hay opciones más económicas, más confiables y más rápidas». Yo creo que ya ha cambiado mucho la percepción del mercado y eso puede ayudar mucho (…) El software libre tiene mucho más cabida en las PyMEs, por el costo de las licencias, son mucho más económicas, porque a comparación de Microsoft, te envuelve en ese juego perverso de: nuevo sistema - el nuevo hardware - el nuevo sistema - el nuevo hardware. Cuando el nuevo sistema corrige en teoría los *bugs* del sistema anterior, cuando ves que las limitaciones de la vieja versión te la sacan en la nueva versión pero te obligan a comprar, no sólo nuevo hardware, nuevas herramientas de desarrollo, entonces se vuelve un juego perverso (…). Por primera vez le pones a una máquina con Windows vista, Windows Seven y corre mejor, por primera vez en la historia, yo creo que por eso le tuvieron que apretar las tuercas [al desarrollo de Windows] (Entrevista 7 a miembro de AMESOL: Marzo 2010).[162]

---

[161] Empresas como Novell o Linspire, han tenido acuerdos con Microsoft para desarrollar interoperabilidad con sus sistemas (Microsoft, 2007b; Novell, 2010a). Del mismo modo Steve Ballmer, actual director ejecutivo de Microsoft, ha hecho patente su intensión de adquirir empresas que se dedican al *open source* (Cowley, 2007).

[162] Otra interpretación de la misma realidad, proviene desde el mundo académico que afirma que de manera consistente con la ley de Brooks, el software decreció en calidad por el hecho de que se trataba de un problema de coordinación en la producción de conocimiento, producto de incrementar la complejidad de la comunicación entre los desarrolladores, al de poner a trabajar más gente en el miso proyecto. “Microsoft fue «salvado» en cierta medida por el extraordinario desarrollo del poder y capacidad de las computadoras personales, - las cuales, crearon un ambiente relativamente comprensivo para software que ha sido pobremente diseñado. También, el vasto crecimiento de las computadoras personales

Esta descripción en extenso, pone de manifiesto la lógica del *lock-in* tecnológico, que no se toma como un "así esta dado". Que a partir de la experiencia del software libre, se ha podido poner de manifiesto qué la industria ha cambiado, pero ¿qué tanto?

En términos de sus estrategias manifiestas, el director de Microsoft, Steve Ballmer, ha calificado la licencia GPL como "un cáncer que se adhiere él sólo, en un sentido de propiedad intelectual, a todo lo que toca" (Ballmer, 2001), puesto que si Microsoft trabajara con software libre, tendría que hacer libre el software que se derive del primero.

En este contexto, se entiende la actitud de esta empresa que fomentan tácticas comerciales que orientan hacia el temor, incertidumbre y duda (FUD; por sus siglas en inglés). En tanto que organizaciones de las que participa Microsoft, como la *International Intellectual Property Alliance* (IIPA), ha manifestado su preocupación por el hecho de que gobiernos minen la competitividad de la industria del software al invertir en proyecto de desarrollo y adopción de software libre. Negando de esta manera, a las compañías privadas acceder al mercado gubernamental (Phipps, 2010). A este tipo de estrategias se le pueden sumar de manera sintética las tácticas en las que ha incurrido esta empresa, tal como lo marca Andrew Oliver (2010):

- No se ha retirado oficialmente su táctica FUD contra Linux
- No se ha pronunciado, como miembro fundador, en contra del intento de la *Business* software *Alliance* y la *IIPA*, de equiparar el software libre con la piratería y como anti-capitalista.
- Continua su ataque, con acciones legales o amenazas, a cualquier código abierto que compita con cualquiera de sus productos principales.
- Continua cooptando las pautas impuestas por "estándares" que se encuentran limitados por patentes o limitaciones en su plataforma .

En este sentido, se puede señalar la campaña publicitaria "*get the facts*"

propició el inicio de usuarios no sofisticados tecnológicamente hablando, quienes no sabían que debían esperar que funcionase mejor, dado que ellos no estaban conscientes de los estándares de confiabilidad, eficiencia y funcionalidad que se había establecido en el software que funcionaba en los *mainframe*" (Weber, 2000, pág. 10 T. del A.). Sin duda una apreciación sólida, que se complementa con el punto de vista de quien fabricaba los chips de computadora, quien debiera tener una excusa para fabricarlos y esta escusa es perfectamente compatible con el argumento que pone de manifiesto esta relación perversa.

iniciada oficialmente en 2004 por parte de Microsoft. La cual, compara principalmente las soluciones de Windows Server y Linux, y que proclama presentar estudios que avalan la superioridad del primero en cuestiones del coste total de propiedad (TCO; por sus siglas en inglés), confiabilidad, seguridad e interoperabilidad. Empero, han existido desconfianzas de la veracidad de dichos estudios, después de que en el año 2002, fuera descubierto el pago de Microsoft a la empresa International Data Corp, encargada del estudio, junto con correo electrónico que pedía "se pusiera a Microsoft bajo una mejor luz" (Foley, 2007a).

Esta campaña recibió mucha crítica, sobre todo porque se encontraba principalmente enfocada en los servicios que ofrecía Red Hat. Dado que dicha empresa no había firmado, como Novell o Linspire[163], acuerdos de patentes con Microsoft, quien clamaba que las distribuciones Linux como la de Red Hat, violaban 235[164] de sus patentes (Foley, 2007b). El 23 de Agosto de 2007, Microsoft retiró su sitio "*get the facts*" y lo remplazó con un nuevo sitio de comparación de soluciones para servidores. En dicha página, siguen disponibles los estudios hechos por "*get the facts*" aunque ya no habrá nuevos estudios "a menos que los clientes los pidan y exista necesidad de mayor información no disponible actualmente entre las investigaciones realizadas" (Foley, 2007d).

Unos días antes (9 de agosto de 2007), esta compañía hizo patente su promesa (hecha en julio de ese año) de poner en proceso aprobación de la OSI, dos licencias: la Microsoft Public License (Ms-PL) y la Microsoft Reciprocal License (Ms-RL) (Foley, 2007c), que fueron aprobadas el 12 de octubre de 2007 (OSI, 2007). Dichas licencias también han sido reconocidas por la Free Software Foundation (FSF) como software libre, aunque recomiendan no hacer uso de ellas, dado que no son compatibles con la GPL[165] (FSF, 2010).

[163] Estas empresas aseguraban que los acuerdos con Microsoft sobre las patentes, "protegían a sus usuarios" de alguna eventual demanda de parte de aquella empresa y que además, los usuarios también ganarían mejor interoperabilidad con sus productos. Puesto que, de sus acuerdos también se derivaban compromisos de cooperación en este rubro.

[164] El problema con este tipo de afirmaciones es que se hacen con base en código cerrado, al que no se tiene acceso y que por lo tanto no se puede comprobar hasta que se revise y compare con el código que según Microsoft violan sus patentes.

[165] Para mayor referencia sobre las implicaciones legales del open source y la compatibilidad

En mayo de 2006, Microsoft puso en funcionamiento un repositorio de proyectos de software de fuente abierta, que hasta principios de enero de 2010, albergaba más de 13 000 proyectos. Sin embargo, este sitio tampoco ha estado excento de críticas, empezando por la icompatibilidad de sus licencias, Ms-PL y Ms-RL, con la GPL, así como el hecho de que alberga proyectos que se proporcionan bajo licencias no aprobadas por la OSI o la FSF y la manera en que controla su junta directiva. Puesto que ésta, se encuentra integrada por empleados de Microsoft (Stallman, 2009; Updegrove, 2009). Donde se teme que Microsoft seduzca a los desarrolladores de software libre, y los reduzca a programar sólo extensiones libres (pero débiles dadas las licencias de Microsoft), que pueden sucumbir ante estrategias como la EEE o demandas por violación de patentes.

De esta forma, y a pesar de todas los detalles mencionados, podemos observar un cambio en las actitudes de Microsoft hacia el software de fuente abierta. De pasar de hablar de él como un cáncer, a lanzar licencias aprobadas por las organizaciones que lo profesan. Así, Microsoft ha liberado 20000 lineas de código para al *kernel* Linux, bajo la GPL (Microsoft, 2009a). Suceso que al principio fue tomado como un acto para congraciarse con la comunidad del software libre (Asay, 2009). A pesar de ello, pronto se descubrió que la liberación se pudo haber debido a que la empresa de open source Vyatta, encontró que los drivers Hyper-V que se liberaron, habían sido creados a partir de código ya disponible bajo la GPL, lo que exponía a Microsoft a una demandada por la violación de los términos de la GPL (Walker-Morgan, 2009).

Existe otro caso, similar al anterior, en el que Microsoft ha tenido que liberar software. La “Windows 7 USB/DVD Download Tool”, que fue retirada del sitio de distribución de Microsoft, bajo sospecha de que se basaba en código liberado bajo la GPL del proyecto “ImageMaster” (S. J. Johnson, 2009b). A esta herramienta le removieron las partes del código que no podía considerarse

con la GPL puede verse Deek & McHugh (2007) especialmente el capítulo VI (págs. 222-264) en su apartado 6.6 (págs. 250-261) que versa sobre cuestiones relevantes de la GPL. Particularmente se puede mencionar que la validez de esta licencia jamás se ha llevado a decidir en algún tribunal de la corte, ya que si alguien incurre en violación de la licencia podría significar una gran perdida de “*good will*” entre las comunidades de programadores y más todavía si se ha llevado a juicio.

como derivado del software previamente liberado y se volvió a poner a disposición del público, bajo la licencia GPL junto con el código fuente (Fried, 2009).

A pesar de estos casos, existe evidencia de que Microsoft, sigue apostando a conformar comunidades de desarrolladores de software libre, a través de la liberación de software como "Web SandBox" (S. J. Johnson, 2009a), la "File Format Software Development Kit (SDK)" y la "PST Data Structure View Tool", todas ellas liberadas bajo la licencia Apache 2.0 (S. J. Johnson, 2010) que es una licencia ampliamente usada por la comunidad de software libre[166], en su sitio CodePlex.

En general, se puede afirma que, esta empresa ha pasado de sólo tratar de comprar a la competencia[167], puesto que no puede ni comprar a todos, ni cooptar así el código libre. Como tampoco ha sido efectiva su táctica de infundir temor a las empresas que se aventuraron con el software libre, y a reprobar a los gobiernos que invierten en software libre, al calificarlos de que atentan contra la propiedad intelectual y la innovación.

Lo anterior, tampoco han tenido la influencia que Microsoft hubiera querido en la industria. Pues aunque han habido empresas que han firmado acuerdos de patentes con él, son las menos ya que su argumento de violación de patentes, debiera empezar por abrir el código para demostrar la similitud con el software libre licenciado bajo la GPL. Donde para casos prácticos, la situación ha sido exactamente la contraria, Microsoft ha usado software sin obedecer los términos de la licencia hasta que ha sido descubierto en flagrancia. Menos aún podría esperar tener éxito con estrategias de *lock-in* o

[166] Aunque se le considera débil, ya que al igual que la licencia BSD, no exige la liberación del código cuando se le han hecho modificaciones al código original. Y por lo tanto, el código resultante puede convertirse en software privativo.

[167] No sólo se trata de la competencia proveniente del open source. Éste es un proceder de cualquier empresa privada. No obstante, a Microsoft se le conoce por haber adquirido varios producto Microsoft, tal es el caso de haber adquirido productos y luego haber hecho grandes ganancias a partir de ellos, como fue el caso de DOS de la Seattle Computer Products, FrontPage de Vermeer Technologies Incorporated, de WebTV lo que ahora se conoce como MSN TV, Hotmail, Direct3D, Internet Explorer de Spyglass, Inc. cuyo proucto era Mosaic, Visio de Visio Corporation y Windows Defender de GIANT Company Software, Inc. que derivó del código que se conocía como GIANT AntiSpyware (Microsoft, 2009b).

EEE, justo para las que licencias como la GPL, fueron diseñadas para evitarla[168].

De esta forma, se puede sostener el argumento de que el cambio de actitud de Microsoft hacia el software libre, se debe al movimiento conjunto del software libre hacia la empresa y de empresas grandes del sector cuando lo integraron en sus modelos de negocios como estrategias antimonopólicas. Así, el tránsito de Microsoft hacia actitudes que favorecen el *open source* no ha sido porque la propia Microsoft haya querido de que el contexto haya permanecido igual, sino como parte de un movimiento doble lleno de luchas, donde si quiere mantenerse ha de tratar adaptar sus estrategias o perder su posición hegemónica.

Sin duda, regresar el tiempo será imposible. Aunque Microsoft de la impresión de que quiera regresar, el movimiento que se ha iniciado continuara. Se trata de una dialéctica cuyo eje es la apropiación del conocimiento, las cuales pueden personificarse y ejemplificarse con la imagen de aquella empresa. Pero sus dinámicas se encuentran en las entrañas del sistema capitalista, que permite cerrar el conocimiento y volverlo un bien escaso de

[168] Estas tendencias parecer estar respaldadas por los resultados del estudio "*2010 predictions research note for open source*", realizada por Gartner Inc., y presentados por Mark Driver (2009). Y cuyo rigor metodológico va más allá, de la "bola de cristal" que su propio título sugiere. Dicho estudio, parte de que para el 2011 "el crecimiento continuo de los que han adoptado el open-source dará por resultado tres categorías distintivas para el software de fuente abierta: (1) proyectos de comunidades, con anchas redes de desarrolladores; (2) proyectos vendo-céntricos, controlados por proveedores comerciales de tecnología; (3) y proyectos comerciales de comunidades, que tendrán vendedores independientes que tienen canales de soporte" (T. del A.). Que sostiene que para el 2012, habrá 90% de probabilidad de que "(...) al menos el 70% de las ganancias del software de fuente abierta comercial provendrán del modelo de negocios de proyectos vendo-céntricos con sistemas duales de licenciamiento" (Driver, 2009 T. del A.), ahora bien, como conclusiones clave, ofrece que: a) El OSS no es anticomercial, pero depende de su éxito comercial; b) Los que adopten de manera conservadora este software, requerirán canales más robustos de soporte. Por otro lado, los que lo han adoptado de manera agresiva, a menudo deben aceptar compromisos con la naturaleza "abierta" del este software y las realidades competitivas con los proveedores comerciales; c) las estrategias más exitosas comercialmente, que han tenido los vendedores de open-source, son aquellas que mezclan elementos de la dinámica del software propietario con los del software de fuente abierta. Las conclusiones que ofrece esta empresa, traen a colación, un hecho que es bien conocido en la industria del software, y es el hecho de que las estrategias de las empresas del software, van más allá de las restricciones de conocimiento para obtener ganancia, y es que en muchas empresas, quien decide qué software se utiliza es el departamento contable, no el departamento técnico. Así, las personas que toman estas decisiones no son los dueños de las empresas, siendo así posible deslumbrarles con prácticas propias del cabildeo para que compren la licencia de uso del software.

manera artificial.

Conclusiones

A través del capítulo IV, hemos podido apreciar una expresión concreta en la forma en la que se viven las prácticas del software libre, desde la perspectiva que brinda su conceptualización como un "bien club". La concreción que se presenta en AMESOL ha surgido de un contexto histórico particular. Dicho contexto, se encuentra definido por las condiciones del mercado que presenta una sociedad capitalista compleja en la que intervienen diferentes sujetos.

En el capítulo anterior, nuestro recorte se había enfocado a la empresa más grande en este sector, cuya efigie personifica la industria privada del software y a tres empresas grandes que forman parte de la AMESOL. En este capítulo nos enfocamos a describir las condiciones que hicieron posible la conformación de la asociación. También, se señaló la importancia que han tenido los incentivos públicos en su formación, pero más allá de eso se hizo énfasis en el sentido que tiene la práctica del software libre para las PyMEs y la influencia de este tipo de prácticas en el panorama general de la industria del software.

Para sostener lo anterior, se describió someramente el contexto histórico en el que se construyó la industria del software en México. Ámbito está marcado por empresas transnacionales, que en primer lugar permitieron la existencia del *hardware* en el país, pero a su vez, alentaron el afianzamiento del modelo de licencias como parte de la sinergia que se estaba gestando como un proceso general. De esta suerte, las crisis y el cambio de modelo económico, permitieron la consolidación de una industria de software en torno a licencias de uso en el ámbito de la microcomputadora. A la vez que entraron transnacionales como Microsoft, SAP y Oracle, en el mercado corporativo. Donde ciertas empresas mexicanas tuvieron un éxito relativo pero insuficiente para poder decir que la industria del software mexicano se consolidó.

La aparición de programas para el desarrollo del software a principios del

nuevo milenio, como el PROSOFT, han producido resultados visibles tanto a nivel regional como nacional. Sin embargo, las PyMEs aún se presentan como las más resegadas, al no poder generar demanda de creación de innovaciones, pues la circulación de conocimiento es prácticamente nula.

La industria del software es dinámica, cambia conforme cambia el software. Con cada innovación cambian las capacidades que él permite desempeñar, el modo de producción, las formas de distribución, las maneras de realización de la ganancia, e incluso, el estilo de consumirlo. Y por todo lo anterior, las diversas empresas se han tenido que adaptar para competir unas con otras.

Justamente en esos momentos históricos, el software libre se consolidó como otra forma de hacer negocios. Ya que técnicamente se afianzó como una opción a través del uso de LAMPP, en los servidores, o bien como plataforma para los ERP's para el sector corporativo. Pero las personalidades que lo sabían manejar eran informales, no sólo en su trato, sino en el trabajo que desempeñaban inherente a la cultura que definieron. En el mundo corporativo si importa el conocimiento, pero también la obediencia a la tiranía de los plazos y la capacidad de replicar en caso de contingencias. Lo que abrió la posibilidad de que empresarios se dedicaran a ofrecer soporte, soluciones, desarrollos en y de software libre.

De esta forma, la AMESOL pudo brindar a estas empresas, la capacidad de acceder a fondos federales y para apoyar su desarrollo. Al constituirse como un organismo intermedio o un organismo promotor, dependiendo de la figura que le corresponda caracterizar a la asociación. Lo cual, también le sirvió como una manera de hacerse de recursos y cubrir los costos operativos de los trámites.

Al interior de la AMESOL existen tres empresas grandes, IBM, Red Hat y Novell y el resto son pequeñas. Empero, aproximadamente la mitad de ellas se acercaron a ella por los recursos federales, más que como una forma de asociarse en torno a un común denominador, el software libre. Ahora bien, de las PyMEs que si se dedican al software libre, se dedican a realizar adaptación

de conocimiento a las necesidades del cliente, puesta a punto, administración y mantenimiento de sistemas.

Esto implica, que existen miembros de la AMESOL que ejercen el software libre como un "bien club". Lo que significa cubrir necesidades en un contexto que le brinda sentido. Dicho contexto, se encuentra conformado por el conocimiento codificado al que tiene acceso a través de Internet, y donde para conferirle valor de uso se requiere adaptarlo a la necesidad particular del cliente, quien lo "consume". De este proceso, puede colegirse un silogismo conformado por el conocimiento tácito que se combina con el conocimiento codificado para solucionar una necesidad, condicionada por la multidimensionalidad de la realidad que implica el contexto.

El acceso a la capacidad de ejercer el software como un "bien club", significa para las PyMEs, el ya no estar supeditadas a un tercero que detenta la legítima restricción del conocimiento. Ofrece una alternativa en la que se encuentran limitados por sus propios conocimientos y capacidades para resolver los problemas técnicos que se les presenta. Esto abre, además, la posibilidad de que las empresas que les "compran" sus desarrollos también puedan hacerlo en el momento que ellos lo decidan. Es decir, que las empresas que adquirieron el software como conocimiento objetivado, también se encuentran en posición de aparenderlo como conocimiento codificado y modificarlo ellos mismos o pedir a alguien más[169] que lo haga por ellos, si no quieren o no se encuentran en posición de formar parte del club capaz de hacerlo.

El conocimiento sobre el software libre le brinda valor de uso, por ello éste es fomentado por las comunidades de conocimiento presentes en Internet. Aquí, la ganancia es posible realizarla porque en el momento que se requiera, existen empresas formales que pueden desempeñar el trabajo, y que además, tienen la capacidad de apropiar el conocimiento que les permita crear la capacidad de resolver problemas específicos. De este modo, haber aprendido a aprender, les brinda la capacidad de poder crear capacidades que creen valor

[169] Donde ese alguien puede o no ser aquel de quien lo han obtenido en primer lugar.

de uso.

Este ejercer el software libre, ha permitido la creación de opciones en una industria propensa al *lock-in*. Lo que conlleva el costo de invertir recursos y tiempo en aprender el código, es decir, existe una barrera dada por el tiempo socialmente necesario para adquirir conocimiento tácito y se capaz de traducirlo en conocimiento explícito. Pero una vez que se tiene aquella capacidad, es necesario mostrar que se dispone del conocimiento a través de las redes de conocimiento mismas. Esto sólo se logrará consolidar como negocio sostenible, si hace que aquel que lo ha contratado considere que le ha proporcionado una solución, no un misterio.

Por otro lado, también se puede realizar la ganancia a través de la enseñanza del conocimiento codificado, lo que crea valor de uso para el software. Aunque su conocimiento es de libre acceso, aún se necesita esfuerzo para apropiarlo, ya sea dedicando tiempo a explorar lo que otros han escrito o bien pagarle a alguien para que comparta lo que sabe.

Al interior de la AMESOL, existe reconocimiento de la característica particular de la historicidad del software libre. Creado en oposición al establecimiento de la restricción del conocimiento vía instrumentos económico-legales. Donde se reconoce que la piratería juega un papel estratégico que no beneficia al software libre.

A través de nuestro análisis, se hace evidente que el software libre ofrece acceso legítimo al software, bajo el mismo régimen legal que legitima su apropiación privada. La forma en que se realiza la apropiación del software libre, permite que se apropie como conocimiento codificado, no cosificado, como lo hace el software privativo. De esta suerte, se puede apreciar un doble movimiento que ha llevado a la legitimación de la industria del software basada en licencias. Y por otro lado, en ese mismo marco de legalidad, se advierte al software libre, como la solución a realizar la cosificación del conocimiento. O dicho de otra manera: el software libre permite la apropiación del software a los usuarios, como conocimiento codificado y, por tanto, le brinda la posibilidad de convertirse en desarrollador, hasta el nivel que le permiten sus propios

conocimientos y recursos económicos, al ejercerlo como un “bien club”.

Donde la práctica de ejercerlo de esta manera, presenta una doble subjetividad, como reconstrucción del pasado (memoria) y apropiación de futuro (utopía). Así, esta práctica es un anudamiento entre el pasado, motivo de creación del distintivo “libre” y, por el otro lado, un futuro deseable, en donde se use un software que se percibe como técnicamente superior. Donde el verdadero plus del software libre, que se ha señalado es la flexibilidad que brinda éste, tanto para sus desarrolladores a la hora de usarlo como herramienta, como para las empresas que lo utilizan para no estar supeditados a cualquier otro ente.

También, ha sido posible articular que la práctica del software libre ha hecho posible cambiar rasgos de la industria que antes se apreciaban como absolutos. Empresas grandes como IBM, Novell y ahora Microsoft han integrado el software libre de una manera u otra. Si bien, las dos primeras lo integraron mucho más abiertamente, en tanto contribuciones en código en diversos proyectos, como en donaciones. El segundo ha experimentado un devenir mucho más accidentado, no porque la propia empresa lo haya querido así, sino porque el propio cambio de la industria se lo ha impuesto.

El movimiento de la industria del software ha sido el producto de tensiones, que son políticas, económicas y sociológicas. La dinámica que hemos observado aquí está dada por la dialéctica entre el mundo del software libre y el software propietario. Donde la industria se encontraba en el *lock-in* que dictaban sólo las grandes empresas que dominaban el mercado. La incursión del software libre en el mercado, lo ha complejizado y ha puesto un nuevo competidor que no se adapta a los cánones bajo los cuales solía aparecer la competencia. Aquí, compartir el conocimiento ha sido su rasgo fundamental, que ha probado ser exitoso para realizar ganancias en un entorno donde la regla era la restricción del mismo.

Así, Microsoft pasó de calificar las licencias como la GPL de “un cáncer”, a pedir que las organizaciones del FLOSS que aprobaran sus propias licencias como software libre. E incluso, a preferir licencias populares dentro del mundo

del software libre, como la Apache License, sobre sus propias licencias.

Además, aquella empresa ha puesto servidores para proyectos *open source.* Sin embargo, sus acciones no dejan de provocar críticas, no sin fundamentos y justificación, por la forma en que la compañía se ha comportado ha través de los años (comportamiento que es propio de toda empresa capitalista). Tal es el caso de haber sido descubierta en flagrante violación de la GPL.

Este devenir, insisto, se ha dado como parte de un movimiento que envuelve a la industria en una dialéctica entre el software privativo y el software libre. Donde en el marco de la sociedad capitalista del conocimiento, el software libre ha estimulado nichos que se encontraban restringidos por la apropiación legítima del conocimiento.

Reflexiones finales

Dado que ya se han presentado conclusiones de cada uno de los capítulos, en esta última parte del cuerpo de la tesis, se propone realizar reflexiones sobre la manera en que se articula cada uno de estos capítulos. Y las significación que esto tiene para la comprensión de la valorización del software libre y la forma en que se realiza ganancia a partir de éste. Lo que ha implicado, que el devenir general de la industria del software, sea presentado como un doble movimiento.

Para lograr esto, partiremos de la forma en que se han respondido los objetivos y las hipótesis de trabajo. En segundo lugar, se abordan los descubrimientos conceptuales que se han hecho en la AMESOL, en donde, también se discuten posibles implicaciones que tiene la confirmación de las hipótesis para esta asociación. En la tercera parte, se hace un recuento de los aportes conceptuales del presente trabajo y los descubrimientos inesperados que ha brindado el trabajo de campo. Por último, se abordan preguntas que han surgido en la presente investigación y que han quedado sin responder, a la espera de quien pueda y quiera contestarlas.

a. Sobre los objetivos y las hipótesis de trabajo

En este apartado, se clarifica la manera en que se han cubierto los objetivos que planteaba esta investigación, así como la forma en que se han resuelto sus hipótesis. De esta suerte, en el siguiente subapartado abordamos el objetivo general y los objetivos específicos que nos hemos planteado. Para luego, en el segundo subapartado, abordar la hipótesis y la forma en la que se han resuelto.

i. En cuanto a los objetivos

La finalidad general de la investigación nos decía:

> Analizar las relaciones entre las nuevas formas de valorización del conocimiento en el proceso de producción del software libre y en el esquema de realización de ganancia de diversas empresas que han incorporado el software libre en sus estrategias de negocios, dedicando atención particular al caso de la Asociación Mexicana Empresarial del Software Libre en su contexto histórico.

Para lograr esto, se estudió: a) la forma en que se consideraba que se realizaba la valorización en otras etapas del capitalismo, y de este modo, lograr la capacidad de distinguir nuevas formas de valorización, que son concretas en el proceso de producción del software. También, b) se hizo una revisión de las estrategias de negocios que tienen las empresas grandes y pequeñas. Igualmente c) se observaron las formas de negocios que orientan a los miembros de la AMESOL y las condiciones de su contexto.

Así, en el primer punto, observamos que a través de las distintas etapas del capitalismo se encontraban diversas formas de restricción del conocimiento. Por lo que se concluyó, que en la sociedad del conocimiento, el mismo conocimiento es un bien escaso de manera legítima pero artificial. Esto último, por los regímenes de propiedad intelectual. Y donde su valorización inmaterial o intangible, se realiza a partir de su transformación de conocimiento tácito en conocimiento útil, mediado por las tecnologías de la información y la comunicación.

En cuanto al segundo punto, observamos que para las empresas grandes que han integrado el software libre como una estrategia, les permite escapara al *lock-in* y la EEE, e incluso acceder a mercados que aparecían como mercados monopolizados. A través de su incursión a través de venta de soporte, soluciones, adaptación, instalación, puesta a punto, consultoría y capacitación, como forma de realizar ganancias.

Por último, se distinguió que entre los miembros de la AMESOL existen miembros que tienen diferentes tamaños y que si existen miembros que se dedican a la práctica del software libre, y más importante aún, que se benefician de las estrategias que también practican las empresas grandes. Además, también se señaló su contexto histórico inmediato, así como el entorno que supusieron diversos apoyos federales, como el PROSOFT en su conformación.

Ahora bien, las metas específicas que se señalaron al inicio de la investigación, eran:

- Analizar las relaciones potenciales entre los principios del software libre y la forma de valorización del conocimiento.

- Analizar las implicaciones de la producción con los principios del software libre desde el ámbito empresarial para la realización de ganancia.
- Estudiar el modo de producción del conocimiento y su apropiación en el esquema general que presentan diversas empresas del software, tanto transnacionales como mexicanas.
- Develar el doble movimiento de la realidad que permitió el proceso de incorporación de nuevas formas de valorización del conocimiento y su repercusión en las formas de realizar ganancia.

Hacer un análisis de las relaciones potenciales entre los principios del software libre y la forma de valorización del conocimiento requirió una conceptualización del software como un conocimiento que se ha hecho concreto. Donde el software libre es susceptible de ser apropiado de dos maneras, la primera y al igual que el software propietario, como conocimiento objetivado, pero también, la segunda, como conocimiento codificado.

Como complemento, y para brindar un panorama de las relaciones que implica la valorización del conocimiento, se revisaron ejemplos de desarrollos concretos de software libre como BSD y Linux; las motivaciones de individuos y empresas para desarrollar y adoptar el software libre; la organización social y la potencialización de los desarrollos técnicos que los hicieron posibles. Así, se observó como los principios de estudio, modificación y redistribución llamaba a los desarrolladores más aventajados que requieren cubrir algún valor de uso. Y cuya praxis acaban por crear software robusto. Esta forma transformación y adaptación del conocimiento (valorización) ha probado fomentar la innovación a través de proyectos que se cuentan por decenas de miles. Todos ellos distribuidos en diversos repositorios disponibles por medio de Internet.

Para acceder a *las implicaciones de los principios del software libre, en cuanto a su forma de producción, para la realización de ganancia desde el ámbito empresarial*, se realizó su conceptualización como un “bien club”. Así, se hace evidente el mecanismo de exclusión, que permite a las empresas realizar ganancias a partir de que aquello se encuentra restringido por las aptitudes y el

tiempo que llevaría apropiarlo. Cuando las empresas, grandes o pequeñas, ejercen este “bien club” les ofrece gran flexibilidad y capacidad de sobrellevar un contexto social internacional contingente, que imponen diversas condiciones a los mercados y, por ende, a la producción. Les permite ser independientes de cualquier otro ente y definir su propio camino a seguir.

En cuanto al *estudiar la forma en que se produce y apropia el software, tanto en transnacionales como en empresas mexicanas*, nos enfocamos en la empresa Microsoft, como paradigma de la forma en como se realizan las prácticas del software privativo. Por otro lado, en el caso del software libre, abordamos a IBM, Red Hat y Novell como ejemplos de grandes empresas transnacionales, y por último abordamos a la AMESOL, como la concreción de empresas mexicanas.

En Microsoft observamos cómo se realiza la concreción del conocimiento en sus metodologías de desarrollo y la realización de la ganancia como si se tratara de un producto tangible. Esto es posible, pues se encuentra protegido técnicamente por tener un código cerrado y legitimado por el régimen jurídico. Lo anterior y su amplia base usuarios, le permite ejercer poder de mercado a través de *lock-in* tecnológico, y estrategias como la EEE. Esto reduce las posibilidades de éxito de productos innovadores en nichos de mercado ya ocupados, sobre todo si siguen metodologías de desarrollo verticales e inflexibles, que asemejan en más de un sentido al taylorismo-fordismo. En cambio, si alguna empresa innova y se inaugura un nuevo nicho del software que tiene posibilidades de crecimiento, esta empresa puede ser adquirida por otra más grande.

De esta manera, observamos que al menos las grandes empresas del software, se encuentran en posición de utilizar el software libre como una estrategia de mercado. En donde se ha utilizado el acceso al conocimiento que permite el software libre, para ejercer su práctica como “bien club” y lograr realizar ganancias a través del soporte y servicios derivados. En este contexto, la adopción del software libre se presenta como una estrategia que permite a los usuarios con altos conocimientos ser capaces de salvar dificultades técnicas

dado el control que tienen del software.

Lo anterior, se hace evidente en el mercado de servidores, donde conocimiento técnico del tema es abundante. O bien, se presenta como estrategia para abrir el mercado en nichos monopolizados y doblemente protegidos, técnica y legalmente. De esta forma, el software libre brinda acceso a herramientas informáticas de alta calidad, sin correr los riesgos de emprender un desarrollo privado en solitario, pues de no tener éxito el conocimiento producido no se pierde ni queda olvidado si ha logrado formar una comunidad de usuarios robusta, pues se trata de conocimiento codificado.

Por último, a lo largo de la tesis, se marcó un *panorama general de un doble movimiento que ha de ser considerado político*, que se presenta como una continuación de las diversas formas de restricción del conocimiento que se han vivido en diversas etapas históricas del propio capitalismo. Esto se debe a que en donde hay relaciones de dominación, hay ejercicio de poder, tal es el caso del ejercicio de poder de mercado que tiene Microsoft. Donde se puede leer la emergencia del software libre como una respuesta política que demostró ser capaz de producir software de alta calidad en el plano técnico y el cambio de una industria en su dimensión económica.

Este proceso se encuentra atravesado por la subsunción del software libre en la lógica del capital, al permitir realizar ganancias como un “bien club” a empresas específicas. Pero también, en general tiene otro efecto para el capitalismo, ya que funciona, a través de su uso, como optimizador de la plataforma de la sociedad del conocimiento, a saber: las comunicaciones y el procesamiento de datos. De esta manera, podemos decir que el proceso político, tiene consecuencias en diversas dimensiones que presentan las contradicciones del capitalismo, lo que le permite incorporarlas. De esta manera, lo que en un nivel se presentan como producto de tensiones (contradicciones y luchas), en la dimensión de su devenir, lo potencian.

ii. Esclarecimiento de las hipótesis

Al cubrir los objetivos, podemos pasar a observar nuestras hipótesis, las cuales

nos dicen:

- Los esquemas tradicionales de valorización del conocimiento se transforman como consecuencia de nuevos avances técnicos y organizaciones productivas resultado de nuevas prácticas sociales.
- La relación entre la valorización del conocimiento y formas de realización de ganancia es dialéctica, por lo que han emergido procesos que expresan la síntesis entre los modelos del software libre y el propietario.

En cuanto a la primer hipótesis, podemos aseverar que los esquemas de valorización del conocimiento se han transformado, puesto que la sinergia del software libre ha impulsado cambios técnicos y productivos. Incluso en empresas como Microsoft, que tendían a seguir patrones jerárquicos e inflexibles.

En cuanto a la segunda, el cuerpo de la tesis pone en evidencia que la relación entre software libre y propietario es dialéctica. En este sentido, el software, en el marco de las sociedades capitalistas en que es producido y utilizado es fundamental la realización de ganancias si es apoyado por empresas. Para lograr ganancias a través del software libre se debe ejercer como un “bien club”, donde las empresas han adoptado y apoyado el desarrollo de software libre porque la barrera de exclusión está dada por la aptitud y el tiempo socialmente necesarios para aprenderlo. Comprender el software lo convierte en conocimiento tácito, que puede ser utilizado para convertir el conocimiento codificado en software libre útil. El común denominador de esto es la transformación del conocimiento en valor de uso, posibilitado en el marco de la exclusión que brinda la pertenencia al club que puede leer el código libremente.

La relación entre la valorización del conocimiento y formas de realización de ganancia es dialéctica, por lo que han emergido procesos que expresan la síntesis entre los modelos del software libre y el propietario. Ahora bien, con esto en mente, en el próximo apartado se señalan de manera más esquemática las diferencias y similitudes que se observan.

b. Los descubrimientos en lo conceptual y en el campo

En este apartado se trata de resumir los descubrimientos que se han hecho. Y que permiten aseverar que se han cubierto las hipótesis. El primer apartado hace un recuento conceptual de los descubrimientos que se hacen, mientras que el segundo, presenta situaciones reales que no se esperaban encontrar,

i. Construcciones conceptuales en su movimiento

Para abordar esta realidad inédita, fue necesario resignificar conceptos, o bien, construir nuevos a partir de los viejos. De esta forma, se conceptualizó el software como conocimiento objetivado y al software libre como conocimiento codificado susceptible de ser ejercido como un “bien club”.

De esta forma, conceptualizar el software como conocimiento codificado permite diferenciar el software propietario del libre. Dado que, únicamente el software libre permite apropiarlo como conocimiento cuando es comprendido. De aquel modo, dentro de la concepción del primero cabe el software privativo, que hace uso de las restricciones legítimas del conocimiento y sólo permite ser apropiado como una cosa. En cambio, el segundo permite ser apropiado como conocimiento codificado.

Ahora bien, a este conocimiento codificado sólo se puede acceder como un “bien club”. Que se encuentra restringido por las aptitudes y el tiempo que llevaría apropiarlo. Esta práctica se ha construido como resultado de la vicisitud histórica del propio software y que se remonta a los inicios del mismo. Y que, dada la estructura del mercado actual ha sido necesario utilizar el “apellido” *libre.*

La práctica del software libre, permite aprovechar cada pieza de código que se encuentra en diversos repositorios. Por lo que se convierte en una biblioteca infinita de programas que potencialmente pueden resolver alguna necesidad que se percibida. De lo que se trata, es de contar con la habilidad y el conocimiento para poder aprovecharlo y resolver necesidades que se presentan como reales, no creadas artificialmente.

Ejercer el software libre como “bien club” permite decidir a empresas,

grandes y pequeñas, sobre su estrategia a seguir y por ello les dota de capacidad de pronta respuesta en un contexto social internacional que imponen diversas condiciones a los mercados y, por ende, a la producción. Lo que lleva aparejado costos, propios de ejercer aquel bien.

Asimismo, se observa la forma en que se integra el trabajado en cada uno de los modelos de empresa, de software libre y el de software propietario, pero sólo como acercamiento conceptual a la realidad. Dado que ésta muestra que poco a poco las empresas se encuentran en caminos que las lleva a la convergencia, el qué tanto se de esto, depende de “luchas” de unos con los otros.

Lo que ha hecho evidente este análisis, es que este movimiento histórico amplio se ha presentado como una relación en la que sujetos realizan prácticas concretas, que cuentan con principios ordenadores propios (los principios del softwae libre, la valorización del software, la realización de ganancia), en busca de sus intereses particulares. Para ello, se requieren formas particulares en que se hacen la relación trabajo-conocimiento, el uso característico de la tecnología y la restricción del conocimiento (Tabla 8).

El movimiento general de ambas formas que invocan tanto las empresas grandes basadas en el software libre, como las que lo hacen en el privativo, por un lado la valorización particular del conocimiento y la forma de realizar su ganancia, ha convergido poco a poco. Unos las han adoptado con más rapidez que otros, dado su posición en el mercado y su interés por mantenerse como dominantes en él (Tabla 9).

Tabla 8: Relación conceptual entre conocimiento, trabajo y tecnología

Organización del trabajo	Relación trabajo -conocimiento	Uso característico de la tecnología	Restricción del conocimiento
Artesanal	El trabajador sabe el propósito de su trabajo y para qué lo hace, en su totalidad.	Tecnologías elementales.	No existe restricción alguna al conocimiento necesario para el trabajo.
Manufactura	Se realiza una cooperación de manera parcelaria y jerárquica.	Selección por habilidades de los trabajadores para utilizar la tecnología.	Se restringe el conocimiento sólo a las tareas asignadas.
Fábrica	El trabajo se encuentra limitado a la exigencia de la máquina.	Uso de máquinas como sujetos principales de la producción.	Existe separación entre el diseño de las máquinas y el conocimiento de uso de la máquina.
Taylorismo-fordismo	Se reduce a sólo las acciones pensadas por y para el trabajador.	Se reduce al trabajador a la cadencia impuesta por la cinta transportadora.	Dada por la división extrema del trabajo, entre agente de la producción y diseño del proceso.
Fabrica post-fordista	Se conforman unidades cooperativas de producción con capacidad de decisión y conocimiento de una parte significativa o de la totalidad del proceso productivo.	Se utilizan modelos flexibles de producción, que requieren dominio de varios tipos de máquinas y procesos automatizados a través de equipos electrónicos.	El conocimiento es normado y estandarizado, ahí donde se requiere comunicación para consumar la cooperación, se realiza valorización inmaterial.

(Continúa)

Tabla 8 *(Continuación)*: Relación conceptual entre conocimiento, trabajo y tecnología

Organización del trabajo	Relación trabajo -conocimiento	Uso característico de la tecnología	Restricción del conocimiento
Empresa red	Delocalización de la producción a nivel planetario requiere conocimiento de los medios, capacidad de aprehender entornos en constante cambio y capacidad de decisión.	Medios de comunicación electrónica, y de procesamiento de información.	Restricciones jurídicas a la producción y apropiación del conocimiento. Se requiere comunicación para consumar la cooperación, se realiza valorización inmaterial.
Empresa de software propietario	Se orienta a estandarizar el proceso de creación de conocimiento, el cual probablemente nuca se repita. Cuyo desarrollo es egocéntrico. Aunque se encuentre geográficamente descentralizado.	Permite que funcione la tecnología de comunicación y procesamiento de información, que es la infraestructura de la sociedad del conocimiento.	Restricciones jurídicas a la producción y apropiación del conocimiento. Se requiere comunicación para consumar la cooperación, pero añade tiempo a subdesarrollo. Se realiza valorización inmaterial, al codificar el conocimiento.
Empresas del software libre	Se basa en colaboración descentralizada, el trabajo se desarrolla en torno a un proyecto. Saber aprender a aprender permite la capacidad general de aprender capacidades para resolver situaciones específicas. Al tiempo que permite generar valor de uso para el conocimiento codificado.	Los inconvenientes de la complejidad son aliviados por avances técnicos como el Internet, la estructura modular y sistemas de control de cambios. Permite que funcione la tecnología de comunicación y procesamiento de información, que es la infraestructura de la sociedad del conocimiento.	La restricción está dada por la aptitud y el tiempo requeridos para acceder al conocimiento (“bien club”).

Fuente: Elaboración propia.

Tabla 9: Diagrama comparativo-conceptual de los proyectos de software libre, las empresas y su convergencia

	Proyectos	**Empresas**	
	Software libre	**Software privado**	**Convergencia: síntesis como producto del devenir**
Valorización	Desarrollo descentralizado. Se hace por contribuciones que potencialmente provienen de todo aquel que encuentre el software útil. Su medio es el internet, herramientas de control de versiones y diseños modulares que parecen mitigar los problemas asociados a la complejidad. Los consumidores son miembros de la comunidad, cuyo principio que guía el desarrollo es satisfacer necesidades.	Desarrollo centralizado. Metodologías institucionalizadas, estandarizadas y jerárquicas. Sujetas a problemas derivados de la división social del trabajo. Donde el incremento lineal de programadores incrementa la complejidad de forma geométrica. Los consumidores no presentan gran retroalimentación.	Desarrollo descentralizado o bien, tiende a incluir a los clientes a través de poner más atención a sus necesidades. Publicación de mejoras y actualizaciones (aunque no en todos los casos) a los proyectos de SL, traducciones, documentación, crear y mantener foros. Puede estar ligado al patrocinio de un proyecto de software libre en particular, del cual es susceptible de utilizar como base de un producto privado.
Licenciamiento	Uso de licencias que permiten el estudio, modificación y redistribución.	Protección legal a partir de la protección legal de derechos de autor o *copyright* y en algunos casos, de patentes.	Cuando son "vendedor único de fuente abierta", se hace uso de ambos tipos de licencias para diferentes partes de sus productos.
Realización de ganancia	-	Venta de licencias de uso. Que se pueden maximizar si a través de la protección técnica y legítima del conocimiento se ejercer poder en el mercado al crear necesidades. Lo cual se logra al realizar estrategias de *lock-in, EEE* e incluso de piratería. Cuando la base de usuarios es amplia y puede considerarse monopólica.	Venta de adaptaciones específicas del software, productos "mejorados", soporte y capacitación. Se evita el *lock-in* y al *EEE*. Hace posible entrar a nichos de mercado ya monopolizados.

Fuente: Elaboración propia.

Otro punto que se puede resaltar de este movimiento, es que el modo de desarrollo del software es descentralizado (tanto geográficamente como en capital). Lo anterior, ha producido la posibilidad de desarrollo fuera de los esquemas empresariales tradicionales, que ha permitido la creación de software que ha llegado a ser considerado como técnicamente superior y seguro por sus propios desarrolladores y usuarios, al compararlo con las opciones propietarias. Donde aquel que tiene los conocimientos lo puede modificar para satisfacer alguna necesidad particular. Esto permite a sus usuarios-desarrolladores, no depender de programas estandarizados que no contemplan este nivel de adaptaciones a la medida (incluidos miembros de AMESOL).

En suma, podemos insistir, que las formas de valorización del conocimiento y realización de la ganancia dentro de la sociedad capitalista del conocimiento, esta dada por la manera en que se restringe este último, y que, su proceso de restricción económico-legal que fomentó reducirlo a conocimiento cosificado, para realizar ganancias de él como si se tratase de cualquier mercancía tangible.

Pero sus características inmateriales o intangibles, han hecho posible compartirlo como conocimiento codificado, susceptible de ser aprendido y por lo tanto, su naturaleza es un circulo cognitivo. Esta última forma de transformación del conocimiento, por lo tanto, que al principio emergió como clara resistencia a los principios que fomentaba el capitalismo, le han probado ser provechosas en más de un sentido. Al proveer plataformas de alta calidad para la infraestructura en su fase cognitiva (o capitalísmo cognitivo), e impulsó a la propia industria del software, a través de competencia, lo que no había acontecido en él bajo la estructura de innovación del capitalismo tradicional.

ii. Descubrimientos en el campo

Las construcciones no esperadas, que devienen del contacto con la realidad estudiada, fueron las siguientes:

- Que el software libre permite a las empresas ser flexibles, ya sea en sus desarrollo o como usuarios, no se encuentran supeditadas a ningún tipo

de *lock-in* tecnológico. Aquí, lo que requieren es: en primer lugar, la habilidad de aprender a aprender, que define la capacidad general de adquirir la capacidad para resolver capacidades específicas. En segundo, la práctica del software libre manifestada como producción/desarrollo y/o adaptación, son las posibilidades que se expresan de manera concreta en realidades particulares. Pero susceptibles de comprensión racional como la valorización del conocimiento en el capitalismo cognitivo.

- También y como consecuencia del anterior, que las capacidades centrales, presentes en las empresas de AMESOL, fuera el aprender a aprende. Que les permite transformar el conocimiento codificado generico en un valor de uso en una situación específica. A este proceso, llámese venta de soluciones, adaptación, instalación, puesta a punto, consultoría, capacitación, en el tiempo y formalidad socialmente aceptable. Esta es la forma de realizar ganancia, de las empresas del software libre en el capitalismo cognitivo. Puesto que, no existe el *lock-in* tecnológico, por lo que en el software libre, podemos hablar de un retorno del conocimiento a quien desempeña el trabajo.
- Que la dimensión subjetiva de las prácticas sociales, sobre todo en empresas de software libre, estuviera relacionada con la existencia concreta de la piratería. Como reconstrucción del pasado (memoria) y apropiación de futuro (utopía). Así, esta práctica es un anudamiento entre el pasado, motivo de creación del distintivo “libre” y, por el otro lado, un futuro deseable, en donde se use un software que se percibe como técnicamente superior. Donde el verdadero plus del software libre, que se ha señalado es la flexibilidad que brinda éste, tanto para sus desarrolladores a la hora de usarlo como herramienta, como para las empresas que lo utilizan para no estar supeditados a una empresa.

En los tres casos anteriores, se advierte que fueron situaciones no esperadas, que fueron recogidas en el marco conceptual que brindaba la presente investigación. Por consiguiente, el primer punto sorprende por el

hecho de que las empresas valoraran como importante la capacidad de resolver los problemas que se les presentan antes que utilizar tecnologías que no tienen manera de saber el cómo funcionan. El segundo punto y que deriva del anterior, es que en las empresas pequeñas como las de AMESOL, exista la capacidad de adquirir capacidades, es decir, de adaptación del conocimiento codificado en un valor de uso para una situación específica. Y por último, que en las empresas, cuyo meta última sea la realización de ganancias, se encuentren presentes consideraciones subjetivas con respecto a lo apropiado de utilizar software que cumple con el marco legal establecido.

c. La AMESOL en los conceptual y en su perspectiva de futuro

Esta asociación, compuesta por 26 miembros registrados, entre las cuales no todas usan software libe. Se encuentran IBM, Red Hat y Novell, como empresas transnacionales y las demás PyMEs. Entre las empresas que si usan software libre, se dedican a realizar adaptación de conocimiento a las necesidades del cliente, puesta a punto, administración y mantenimiento de sistemas.

Dados los resultados que se han manejado antes, y a que la asociación se encuentra configurada para apoyar a las PyMEs como un organismo promotor, haremos énfasis en cuanto a este tipo de empresas. Por esto, a continuación pasamos a ver las empresas de AMESOL de acuerdo a los conceptos ordenadores que nos hemos planteado desde el principio. Y en segundo plano, abordaremos la perspectiva de futuro que puede tener la AMESOL si los delineamientos que apuntan las hipótesis se cumplen.

i. Las conclusiones en tanto valorización y realización de la ganancia

El cómo se realiza aquí la valorización de conocimiento y realización de la ganancia tiene que ver, con que el primero deviene del ejercicio del software libre como conocimiento codificado, al que se tiene acceso a través de Internet y que al adaptarlo a la necesidad particular del cliente, quien lo “consume”, se le confiere valor. Del proceso anterior, se puede colegir un silogismo conformado

por el conocimiento tácito que se combina con el conocimiento codificado para solucionar una necesidad, dado por la multidimensionalidad de la realidad que implica el contexto.

La realización de la ganancia es posible, porque el software libre es un "bien club", definido por un mecanismo de exclusión, a saber: el tiempo socialmente necesario para adquirir la capacidad de apropiar el conocimiento codificado en el software libre como conocimiento tácito. Así, la valorización se presenta al pensar el software como conocimiento codificado, mientras que la realización de la ganancia es posible al pensarlo como conocimiento objetivado.

De acuerdo a nuestro análisis, las PyMEs que componen la AMESOL, no están supeditadas a quien detenta la legítima restricción del conocimiento, sino por sus propios conocimientos y capacidades para resolver los problemas técnicos que les presentan y que abren, además, la posibilidad de que las empresas que les compran sus desarrollos también puedan hacerlo en el momento que ellos lo decidan.

De esta manera, las PyMEs, como las que conforman la AMESOL, pueden adquirir, usar, adaptar, cambiar y crear, conocimiento para contextos específicos en un momento particular. Pues son capaces de hacerse del conocimiento a través de redes de conocimientos, tanto codificado como de transformarlo en valor de uso, siempre y cuando tengan las habilidades que lo permitan ("bien club").

Por último, podemos mencionar que son las prácticas sociales de empresas como la que conforman AMESOL, las que han habilitado la propia existencia de estos cambios del en las estrategias de realización de ganancia en la industria del software. La concreción de la transformación del conocimiento de nuevas formas, es decir, de valorizarlo, se ha hecho posible en el capitalismo cognitivo.

De esta suerte, podemos decir que las empresas que conforman la AMESOL, son una expresión concreta del devenir que compone la textura de la realidad que es la sociedad que vivimos día a día. Marcado por un doble movimiento entre el software privativo y el libre, una dialéctica que ha

subsumido procesos que lo ha negado al inicio, pero que al final lo ha potencializado.

ii. Si las hipótesis se cumplen

A pesar de aquello, en este apartado veremos que ante los resultados generales que enmarcan las hipótesis, se deriva que la industria se encuentra convergiendo. Donde la realidad que este fenómeno delinea, a saber: se muestra que cada vez más el software libre tiene participación de la realización de plusvalor, cabe preguntar ¿qué sentido tendrá AMESOL en un futuro cercano? Si es que la tendencia de la industria en general es de cada vez más incluir software libre, como parte de las entre las estrategias del mercado.

Si se es rígido en la interpretación de la realidad, parecería que la AMESOL estaría destinada a desaparecer. Ya que toda empresa que maneje software libre, como usuario, como estrategia de mercado, como herramienta de adaptación de conocimiento, como bloque de construcción de aplicaciones específicas para situaciones particulares, es decir, realizar ganancia a partir de la creación de un valor de uso, etc. sería candidata a formar parte de la AMESOL, en donde hasta Microsoft podría aspirar a formar parte de ella.

Adempero, tenemos que observar que la realidad es compleja, y que si esta asociación tiene sentido para las PyMEs, entonces la AMESOL tiene sentido. No debemos olvidar que la razón principal por la que se han asociado es el trámite de recursos federales, derivados de políticas públicas al sector (PROSOFT) o bien, para el fomento de las PyMEs.

Sin embargo, lo que si es evidente, es que tiene un fuerte componente ideológico, que se deriva de la propia historicidad del software libre. En este sentido, tal vez convendría a la propia asociación ejercer un control mayor sobre sus lineamientos para aceptar nuevos miembros, para que al menos sólo las empresas que manejan software libre se encuentren representadas. Ya sea que se identifican con la ideología, motivaciones o prácticas que hacen posible el software libre, y que aquellas que no se han inscrito en ella pero comulgan con lo anterio la encuentren atractiva.

d. Futuros temas de investigación

En un trabajo de ésta naturaleza, siempre aparecen hilos apasionantes de ser investigados. Pero a los que uno debe dejar para un momento posterior, o bien, para que otros continúen y reelaboren sobre las reflexiones propias y sobre una realidad que ya habrá cambiado. A continuación se listan algunos temas que se han identificado, pero que por falta de recursos, tiempo e integridad de la tesis se han dejado de lado.

En primer lugar, podemos mencionar, la aproximación de la formación de capacidades en las empresas, es un tema muy rico, que podría aportar mucho más sobre la forma y las tipologías de innovación que se pueden producir en las PyMEs. Como parte de procesos de creación y adaptación del conocimiento, que se aprende y aplica en ellas, sin embargo, esta perspectiva va más allá de lo que se puede hacer con la extensión y posibilidades actuales, en recursos material, de quien ha escrito esta indagación. Se trata de un tema interesante, que se deberá dejar para posteriores investigaciones.

Otro análisis que se vislumbra excitante, es el de las relaciones que se pueden apreciar entre el gobierno y las empresas con respecto al software libre. Y como su presencia puede o no significar las prácticas concretas de unos y otros frente a la piratería, no sólo de palabra, sino de acciones. En este sentido, cabría hacer una cuantificación del software libre tanto en empresas como el que se produce y utiliza en el sector público y que escapa totalmente a las condiciones materiales de un sólo estudiante de postgrado.

El devenir ha hecho del software libre, un elemento que ha marcado la textura de la realidad en que vivimos, por esta razón, existen multitud de aristas desda las cuales, resulta emocionante conocer, tal es el caso de las motivaciones a nivel micro. O bien, la forma en que se generan las capacidades en empresas grandes, las cuales son responsables de otro aspecto social importante para la sociedad contemporánea, como es el trabajo que estas producen. En este sentido falta ver cómo inciden directamente las políticas públicas (como el PROSOFT) en el sector y en el desarrollo de las PyMEs que se dedican al software libre (cuyo universo no se encuentra totalmente

representado por la AMESOL).

Por último, no me queda más que esperar tener futuro, con capacidad y los recursos necesarios para continuar con la investigación de estos temas sociales, que han generado en mí el interés por investigarlos y comprenderlos en su justa dimensión e implicaciones.

Referencias

Alonso, L. E. B. (1998). *La Mirada Cualitativa En Sociología: Una Aproximación Interpretativa*. Ciencia. Madrid: Editorial Fundamentos.

Andrade, L. (2007). Del tema al objeto de investigación en la propuesta epistemológica de Hugo Zemelman. *Cinta Moebio*, *30*, 262-286.

Asay, M. (2009, Junio 20). Microsoft embraces GPL, opens Hyper-V to Linux with LinuxIC. *The Open Road - CNET News*. Recuperado Mayo 31, 2010, a partir de http://news.cnet.com/8301-13505_3-10290686-16.html

Aslett, M. (2008, Junio 20). Applying the Bee Keeper model beyond captive open source projects. *451 CAOS Theory*. Recuperado Abril 8, 2010, a partir de http://blogs.the451group.com/opensource/2008/06/20/applying-the-bee-keeper-model-beyond-captive-open-source-projects/

Aspel. (2004). Aspel. *¿Quiénes Somos?*. Recuperado Mayo 4, 2010, a partir de http://www.aspel.com.mx/mx/rec/acerca/quienes.html?idsa=

Ballmer, S. (2001, Junio 1). Microsoft CEO takes launch break with the Sun-Times. Recuperado a partir de http://web.archive.org/web/20010615205548/http://suntimes.com/output/tech/cst-fin-micro01.html

Barahona, I. D., & Alonso, I. (2007). *Historia de la Iinformática en México: 1959-2003* (Licenciatura). México, D.F: Fundación Arturo Rosenblueth.

Beck, K., & Beedle, M. (2001). Manifesto for Agile Software Development. Recuperado Abril 6, 2010, a partir de http://www.agilemanifesto.org/

Bednarz, A. (2010, Marzo 1). Novell CEO's 2009 pay valued at $5.7 million. Recuperado Abril 9, 2010, a partir de http://www.networkworld.com/news/2010/030110-novell-ceo-compensation.html

Beltrán, M. (1991). *La realidad social*. Barcelona: Tecnos.

Benkler, Y. (2002). Coase's Penguin, or, Linux and The Nature of the Firm. *The Yale Law Journal*, *112*(I3), 369–446.

Berlecon, R. G. (2002). *FLOSS report. Free/Libre Open Source Software: Survey and Study*. Final Report. Berlin: International Institute of Infonomics. Recuperado a partir de http://www.flossproject.org/report/

Birchfield, V. (1999). Contesting the Hegemony of Market Ideology: Gramsci's 'Good Sense' and Polanyi's 'Double Movement'. *Review of International Political Economy*, *6*(1), 27-54.

Blender, F. (2009, Julio). blender.org - History. Recuperado Mayo 27, 2010, a partir de http://www.blender.org/blenderorg/blender-foundation/history/

Block, F., & Polanyi, K. (2003). Karl Polanyi and the Writing of "The Great Transformation". *Theory and Society*, *32*(3), 275-306.

Bodas, D. J. (2003, Junio 27). El CMM y la mejora continúa del proceso de software. *willy.net Ingeniería Técnica Informática de Sistemas CES Felipe II (UCM)*. Recuperado a partir de http://www.computing-es.com

Borja, R. (2002). *Enciclopedia de la política H-Z* (3º ed., Vols. 1-2, Vol. 2). México: Fondo de Cultura Económica.

Brooks, F. P. (1995). *The mythical man-month*. Addison-Wesley Pub. Co.

Buchanan, J. M. (1965). An Economic Theory of Clubs. *Economica*, New Series, *32*(125), 1-14.

Buenrostro, E. (2010). Innovación Colectiva, Beneficios Privados: El Software de Código Abierto. *RICEC – Red de Investigacion InterContinental sobre la Economia /la Sociedad del Conocimiento*, *2*(1). Recuperado a partir de http://ricec.info/images/stories/articlerevue/volume2_N1/article/IRICEC3_-_BUENROSTRO.pdf

Cano, A. (2006, Noviembre 25). *Apropiación de la Virtualización del Conocimiento: Producción, Distribución y Consumo de una expresión de desigualdad. Caso: La piratería en la ciudad de puebla, genealogía y contexto* (Licenciatura). Puebla: Benemérita Universidad Autónoma de Puebla. Recuperado a partir de http://pdccpirateria.awardspace.com/INICIO.htm

Canonical. (2010, Julio 27). The Ubuntu Forum Community. *Ubuntu Forums*. Recuperado Julio 28, 2010, a partir de http://ubuntuforums.org/

Casalet, M. (1995). Una Nueva Orientación en la Relación Innovación-Producción en México. *Perfiles latinoamericanos, Revista de la Facultad Latinoamericana de Ciencias Sociales Sede México*, *4*(7), 99-119.

Casalet, M. (2000). Innovación. En L. B. Olamendi (Ed.), *Léxico de la política* (págs. 344-351). México, D.F: Fondo de Cultura Económica.

Casalet, M. (2008). *La oportunidad de los clusters de software para el desarrollo de nuevas competencias y vínculos entre la universidad y las*

empresas. México: FLACSO sede México.

Casalet, M., Buenrostro, E., & Becerril, G. (2008). La construcción de las redes de innovación en los cluster de software. En *Transferencia del Conocimiento y la Tecnología: reto en la Economía Basada en el Conocimiento* (págs. 1-26). Presented at the Congreso Internacional de Sistemas de Innovación para la Competitividad Tercera Edición, León, Guanajuato: QUIVERA/CONCYTEG. Recuperado a partir de http://octi.guanajuato.gob.mx/sinnco/formulario/MT/MT2008/MT2/SESION2/MT2_CASALET_BUENROSTRO_BECERRIL.pdf

Castells, M. (2001). *La galaxia Internet*. Barcelona: Plaza y Janés.

Castells, M. (2002). *La era de la información: economía sociedad y cultura. La sociedad red*. (C. M. Gimeno, Trad.) (4º ed., Vols. 1-3, Vol. 1). México: Siglo veintiuno editores.

Ceballos, D. (2009). Bienvenido al sitio de AMESOL. — Sitio de AMESOL. Recuperado Noviembre 27, 2009, a partir de http://amesol.v.sandino.net:8888/ploneamesol/

CGI. (2010). *Pesquisa sobre o Uso das Tecnologias da Informação e da Comunicação no Brasil 2009*. São Paulo: CETIC.BR. Recuperado a partir de http://www.cetic.br/

Coriat, B. (2005). *El taller y cronómetro* (14º ed.). México: Siglo veintiuno editores.

Corona, E. (2009). IT Complements “salta” del gobierno al corporativo. *IT Complements*. Recuperado Mayo 4, 2010, a partir de http://www.itcomplements.com/esp/noticias/noticia2b.html

Coulais, A. (2009, Septiembre 28). PC-Windows bundling, first hearing at TGI in Paris : UFC Que Choisir against Darty. *AFUL*. Recuperado Abril 9, 2010, a partir de http://aful.org/communiques/court-paris-ufc-darty

Cowley, S. (2007, Octubre 18). Ballmer: Microsoft Will Buy Open-Source Companies. *The Linux Channel - IT Channel News And Views by CRN and VARBusiness*. Recuperado Mayo 31, 2010, a partir de http://www.crn.com/software/202404305;jsessionid=41HOHO3442HHBQE1GHRSKHWATMY32JVN

Curtis, K. (2010). Linux. *keithcu.com*. Recuperado Julio 25, 2010, a partir de http://keithcu.com/wordpress/?page_id=599

CXO Media. (2010). Adobe Speaks Out on Microsoft PDF Battle. *CIO.com - Business Technology Leadership*. Recuperado Abril 7, 2010, a partir de

http://www.cio.com/article/22058/Adobe_Speaks_Out_on_Microsoft_PDF_Battle

Dalle, J., David, P. A., Ghosh, R. A., & Steinmueller, W. E. (2004). *Advancing Economic Research on the Free and Open Source Software Mode of Production* (No. 04-03) (págs. 1-24). Stanford: Stanford Institute for Economic Policy Research. Recuperado a partir de http://ideas.repec.org/p/wpa/wuwpio/0502007.html

Daum, J. H. (2003). *Intangible assets and value creation*. J. Wiley.

Debroy, B., & Chakraborty, D. (2007). *Anti-dumping: global abuse of a trade policy instrument*. Academic Foundation.

Deek, F. P., & McHugh, J. A. (2007). *Open Source Technology and Policy*. Cambridge University Press.

De Laat, P. B. (2005). Copyright or copyleft?:: An analysis of property regimes for software development. *Research Policy*, *34*(10), 1511–1532.

DistroWatch. (2010, Julio 27). Put the fun back into computing. Use Linux, BSD. *DistroWatch.com*. Recuperado Julio 28, 2010, a partir de http://distrowatch.com/index.php?dataspan=1

Dix, M. (2008, Abril 3). Microsoft Announces Extended Availability of Windows XP Home for ULCPCs: Q&A: Michael Dix, General Manager of Windows Client Product Management, discusses Microsoft's commitment to deliver Windows to customers for a new category of devices known as ultra low-cost personal computers (ULCPCs). Recuperado Abril 9, 2010, a partir de http://www.microsoft.com/presspass/features/2008/apr08/04-03xpeos.mspx

Dixon, J. (2009). *The Bees and the Trees: The Beekeeper Model of Commercial Open Source Software*. Pentaho: Open Source Bussines Intelligence. Recuperado a partir de http://www.pentaho.org/beekeeper

Driver, M. (2009, Diciembre 8). Open Source Predictions For 2010. *2010 Gartner, Inc*. Recuperado Mayo 24, 2010, a partir de http://blogs.gartner.com/mark_driver/2009/12/08/open-source-predictions-for-2010/

Ferroni, M., & Mody, A. (2004). Incentivos Globales en Bienes Públicos Internacionales: Introducción y Visión General. En *Bienes públicos internacionales: Incentivos, Medición y Financiamiento*, Economía Internacional (págs. 1-28). México, D.F.: Banco Interamericano de Desarrollo y Alfaomega Colombiana S. A. Recuperado a partir de http://www-

wds.worldbank.org/external/default/WDSContentServer/WDSP/IB/2005/08/08/000011823_20050808110237/Rendered/PDF/248841bienes0publicos.pdf

Finkle, J. (2007, Julio 26). Dell to expand Linux PC offerings, partner says. Recuperado a partir de http://www.reuters.com/article/technologyNews/idUSN2622292420070726

Flynn, I. M., & Mchoes, A. M. (2001). *Sistemas operativos*. Cengage Learning Editores.

Fogel, K. (2007). *Producir Software de Código Abierto*. Stanford, California: Attribution-ShareAlike. Recuperado a partir de http://producingoss.com/es/index.html

Foley, M. J. (2007a, Febrero 2). Microsoft tried to muck with anti-Linux 'facts'. *ZDNet*. Recuperado Mayo 31, 2010, a partir de http://www.zdnet.com/blog/microsoft/microsoft-tried-to-muck-with-anti-linux-facts/235

Foley, M. J. (2007b, Junio 21). Now three Linux vendors won't sign patent deals with Microsoft. *ZDNet*. Recuperado Mayo 31, 2010, a partir de http://www.zdnet.com/blog/microsoft/now-three-linux-vendors-wont-sign-patent-deals-with-microsoft/529

Foley, M. J. (2007c, Agosto 13). Microsoft Shared Source licenses are now in OSI's hands. *ZDNet*. Recuperado Mayo 31, 2010, a partir de http://www.zdnet.com/blog/microsoft/microsoft-shared-source-licenses-are-now-in-osis-hands/638?tag=mantle_skin;content

Foley, M. J. (2007d, Agosto 23). Microsoft kills its 'Get the Facts' anti-Linux site. *ZDNet*. Recuperado Mayo 31, 2010, a partir de http://www.zdnet.com/blog/microsoft/microsoft-kills-its-get-the-facts-anti-linux-site/670

Fried, I. (2009, Diciembre 9). Microsoft reposts Windows 7 download tool. *Beyond Binary - CNET News*. Recuperado Mayo 26, 2010, a partir de http://news.cnet.com/8301-13860_3-10412826-56.html?part=rss&subj=news&tag=2547-1001_3-0-5

Free Software Foundation (FSF). (2010, Abril 26). Various Licenses and Comments about Them. *GNU Project - Free Software Foundation (FSF)*. Recuperado Mayo 31, 2010, a partir de http://www.gnu.org/licenses/license-list.html

Gallagher, D. (2010, Mayo 10). Android market share passes iPhone's: NPD

data. *MarketWatch*. Recuperado Julio 27, 2010, a partir de http://www.marketwatch.com/story/android-market-share-passes-iphones-npd-data-2010-05-10

García-Arias, J. (2004). Un nuevo marco de análisis para los bienes públicos: La Teoría de los Bienes Públicos Globales. *Estudios de Economía Aplicada*, *22*(2), 187-212.

Gates, W. H. (1976, Febrero 3). An Open Letter to Hobbyists. Recuperado Noviembre 25, 2009, a partir de http://www.comcreations.com/os/section_who/gates_letter.htm

Geeknet, I. (2010). About. *Sorceforge: Find and develop open source software*. Recuperado Mayo 13, 2010, a partir de http://sourceforge.net/about

Gorz, A., & Marx, K. (1977). *Crítica de la división del trabajo*. Barcelona: Laia.

Granneman, S. (2003, Octubre 2). Linux vs. Windows Viruses. *SecurityFocus*. Recuperado Mayo 30, 2010, a partir de http://www.securityfocus.com/columnists/188

Hamblen, M. (2009, Octubre 6). Android to grab No. 2 spot by 2012, says Gartner. *Computerworld*. Recuperado Julio 27, 2010, a partir de http://www.computerworld.com/s/article/9139026/Android_to_grab_No._2_spot_by_2012_says_Gartner

Hawkins, R. E. (2002). The Economics of Open Source Software for a Comptetitive Firm: Why give it away for free. *Kluwer Academic Publishers*, 1-18.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research policy*, *32*(7), 1159–1177.

Himanen, P., & Castells, M. (2002). *La ética del hacker y el espíritu de la era de la información*. España: Ediciones Destino.

Hormby, T. (2006, Septiembre 25). VisiCalc and the Rise of the Apple II. *Before the Macintosh*. Recuperado Mayo 4, 2010, a partir de http://lowendmac.com/orchard/06/visicalc-origin-bricklin.html

Hümmer, T. (2010, Febrero). International OpenOffice market shares. *News: International OpenOffice market shares - Portal - Tutorials, Tipps und Tricks für Webmaster auf Webmasterpro.de*. Recuperado Mayo 27, 2010, a partir de http://www.webmasterpro.de/portal/news/2010/02/05/international-openoffice-market-shares.html

IBM. (2005, Noviembre 2). Open Source Software. *IBM Systems Journal*, *44*(2). Recuperado a partir de http://www.research.ibm.com/journal/sj44-2.html

IBM. (2008). Infrastructure simplification and business integration—highly secure, cost-effective and easy to manage: Linux on IBM System z10. IBM. Recuperado a partir de ibm.com/systems/z/linux

IBM. (2009, Septiembre 10). IBM & the Linux Community: Supporting the community development of Linux since 1999. *IBM*. CT556, . Recuperado Abril 9, 2010, a partir de http://www-03.ibm.com/linux/community.html

Instituto Nacional de Estadística Geografía e Informática (INEGI), Comisión Federal de Electricidad (CFE), & Museo Tecnológico (MUTEC). (2003). Grandes Momentos del Cómputo en México 1958-2003. Recuperado Mayo 4, 2010, a partir de http://www.cuentame.inegi.gob.mx/museo/gm/

Johnson, B. (2009, Octubre 21). Mitch Kapor: looking beyond the open source battle. *Technology | The Guardian*. Recuperado Noviembre 29, 2009, a partir de http://www.guardian.co.uk/technology/2009/oct/21/mitch-kapor-open-source

Johnson, S. J. (2009a, Enero 28). Microsoft Puts 'Web Sandbox' Into Open Source. *InternetNews.com*. Recuperado Mayo 26, 2010, a partir de http://www.internetnews.com/dev-news/article.php/3799446/Microsoft-Puts-Web-Sandbox-Into-Open-Source.htm

Johnson, S. J. (2009b, Noviembre 10). Microsoft Nixes Windows 7 Tool on GPL Concerns. *InternetNews.com*. Recuperado Mayo 26, 2010, a partir de http://www.internetnews.com/software/article.php/3847886/Microsoft-Nixes-Windows-7-Tool-on-GPL-Concerns.htm

Johnson, S. J. (2010, Mayo 25). Microsoft Makes Outlook Tools Open Source. *Datamation.com*. Recuperado Mayo 26, 2010, a partir de http://itmanagement.earthweb.com/osrc/article.php/3884041/Microsoft-Makes-Outlook-Tools-Open-Source.htm

Jones, P., & Erwan, H. (2010, Julio 25). Novell-SCO Timeline. *Groklaw*. Recuperado Julio 27, 2010, a partir de http://www.groklaw.net/staticpages/index.php?page=20040319041857760

Karels, M. J., & Smith, C. F. (1998, Marzo 17). Installing and Operating 2.9BSD. *2.9BSD setup*. Recuperado Mayo 6, 2010, a partir de http://minnie.tuhs.org/PUPS/Setup/2.9bsd_setup.html

Kavanagh, P. (2004). *Open source software*. Burlington: Digital Press.

Kretschmer, M. (2003). Software as Text and Machine: The Legal Capture of Digital Innovation. *Electronic Law Journals*. Recuperado Febrero 25, 2010, a partir de http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2003_1/kretschmer/

Lai, E. (2009, Noviembre 4). Linux's share of netbooks surging, not sagging, says analyst. Recuperado Abril 9, 2010, a partir de http://www.computerworld.com/s/article/9140343/Linux_s_share_of_netbooks_surging_not_sagging_says_analyst

Lancanshire, D. (2001, Diciembre). The Fading Altruism of Open Source Development. Recuperado Mayo 10, 2010, a partir de http://pascal.case.unibz.it/retrieve/3262/index2.html

Larson, C. (2007, Julio 3). OOXML and VML, reject. Recuperado Abril 20, 2010, a partir de https://forums.scc.ca/forums/scc/dispatch.cgi/public/viewNextDoc/100285

Lazzarato, M. (1997a). Trabajo inmaterial y subjetividad (con Antonio Negri). (R. Sánchez Cedillo, Trad.). Recuperado Noviembre 16, 2009, a partir de http://www.brumaria.net/textos/Brumaria7/03mauriziolazzarato.htm

Lazzarato, M. (1997b). Trabajo inmaterial y subjetividad. (R. Sánchez Cedillo, Trad.). Recuperado Noviembre 16, 2009, a partir de http://www.brumaria.net/textos/Brumaria7/03mauriziolazzarato.htm

Lazzarato, M. (2000). El «ciclo» de la producción inmaterial. Recuperado Octubre 5, 2009, a partir de http://www.sindominio.net/contrapoder/article.php3?id_article=13

Leonard, A. (2000, Mayo 16). BSD Unix: Power to the people, from the code. How Berkeley hackers built the Net's most fabled free operating system on the ashes of the '60s -- and then lost the lead to Linux. *Salon Free Software Project*. Recuperado Julio 26, 2009, a partir de http://www.salon.com/tech/fsp/2000/05/16/chapter_2_part_one/index3.html

Lerner, J., & Tirole, J. (2000, Febrero). The Simple Economics of Open Source. Harvard Business School's California Research Center.

Lessig, L. (1999). *Code and other laws of cyberspace*. New York: Basic Books.

Lessig, L. (2002). *The future of ideas: the fate of the commons in a connected world*. New York: Randome House.

Lessig, L. (2004). *Free Culture*. New York: The Pinguin Press.

Licona, A., Angoa, S., Victoria, J., Bautista, R., & Burgos, M. (1985). El Software para las Microcomputadoras. Recuperado Mayo 3, 2010, a partir de http://delta.cs.cinvestav.mx/%7Emcintosh/comun/historiaw/node31.html

López, F. J. (2010, Abril). Francisco J. Lopez / Curriculum Vitae. *Curriculum Vitae*. Recuperado Mayo 4, 2010, a partir de http://facultyweb.maconstate.edu/francisco.lopez/resume.html

Luna, E. O., & Taritolay, D. (2004, Junio 28). Bienes intangibles.

Martens, C. (2007, Mayo 8). JAVAONE: Sun - The bulk of Java is open sourced. *ITworld*. Recuperado Mayo 29, 2010, a partir de http://www.itworld.com/070508opsjava?page=0%2C0

Marx, C. (1984). *Cuaderno tecnológico-histórico. Estudio Preliminar de Enrique Dussel*. Puebla: Universidad Autónoma de Puebla.

Marx, K. (2002). *Elementos fundamentales para la crítica de la economía política: Grundrisse, 1857-1858*. (J. Aricó, P. Scaron, & M. Murmis, Trads.) (11º ed., Vol. 1). Siglo XXI.

Marx, K. (2004). *El Capital, Libro primero: El proceso de producción del Capital* (Vol. 2). México: Siglo XXI.

Marx, K., & Engels, F. (1983). La tecnología del capital. Subsunción Formal y Subsunción Real del Proceso de Trabajo al Proceso de Valorización (Extractos del Manuscrito 1861 1863). En B. Echeverría (Trad.), *Cuadernos Políticos*. México: ERA. Recuperado a partir de http://www.bolivare.unam.mx/traducciones/subsuncion.html

McConnell, S. (2008, Marzo 31). Chief Programmer Team Update - 10x Software Development. (Del autor, Trad.). Recuperado Marzo 23, 2010, a partir de http://forums.construx.com/blogs/stevemcc/archive/2008/03/31/chief-programmer-team-update.aspx

McPherson, A., Proffitt, B., & Hale-Evans, R. (2008, Octubre). Whitepaper: Estimating the Total Development Cost of a Linux Distribution. *The Linux Fundation*. Recuperado Noviembre 8, 2009, a partir de http://www.linuxfoundation.org/publications/estimatinglinux.php

Microsoft. (2007a). Microsoft Fourth Quarter FY 2007 Earnings Release. Recuperado Abril 7, 2010, a partir de http://www.microsoft.com/msft/earnings/FY07/earn_rel_q4_07.mspx

Microsoft. (2007b, Junio 13). Microsoft and Linspire Collaboration Promotes

Interoperability and Customer Choice: Broad agreement will facilitate interoperability between Windows and Linux, provide intellectual property assurance. *Microsoft News Center*. Press, . Recuperado Mayo 31, 2010, a partir de http://www.microsoft.com/presspass/press/2007/jun07/06-13LinspirePR.mspx

Microsoft. (2009a, Junio 20). Microsoft Contributes Linux Drivers to Linux Community: Roundtable Q&A: Sam Ramji, senior director of Platform Strategy at Microsoft, and Tom Hanrahan, director of Microsoft's Open Source Technology Center, discuss the company's release of Linux device driver code under General Public License v2. *Microsoft News Center*. Press, . Recuperado Mayo 31, 2010, a partir de http://www.microsoft.com/presspass/features/2009/jul09/07-20linuxqa.mspx

Microsoft. (2009b, Diciembre). Acquisitions. *Microsoft Investor Relations*. Recuperado Julio 28, 2010, a partir de http://www.microsoft.com/msft/acquisitions/history.mspx

Mochi, P. Ó. (2004). La Producción de Software, Paradigma de la Revolución Tecnológica. En *Sociedad de la información y el conocimiento, entre lo falaz y lo posible* (págs. 313-343). México: La crujía ediciones.

Mochi, P. Ó. (2006). *La industria del software en México en el contexto internacional y latinoamericano*. Cuernavaca, Morelos, México: UNAM. Recuperado a partir de http://132.248.9.9/libroe_2006/0003859/

Mochi, P. Ó., & Hualde, A. (2006). *La industria del software en México: Informe* (Final). Oportunidades y desafíos de la industria del Software en América Latina. México: CEPAL.

Mochi, P. Ó., & Hualde, A. (2008, Mayo). México: ¿una apuesta estratégica por la industria del software? *Comercio Exterior*, *58*(5), 335-349.

Mochi, P. Ó., & Hualde, A. (2009). México: Producción Interna e Integración Mundial. En P. Bastos & F. Silveira (Eds.), *Desafíos y Oportunidades de la Industria del Software en América Latina* (págs. 171-203). Colombia: CEPAL y Mayol Ediciones. Recuperado a partir de http://www.eclac.cl/publicaciones/xml/5/35655/Capitulo6.pdf

Mochi, P. Ó., & Rivera, M. Á. (2006). Nueva modalidad de desarrollo y tecnologías de la información: El caso del software. Avances de un sexenio, 2000-2006. *Economía Informa*, *343*.

Mondok, M. (2007, Marzo 12). Microsoft executive: Pirating software? Choose Microsoft! Recuperado Mayo 29, 2010, a partir de http://arstechnica.com/microsoft/news/2007/03/microsoft-executive-

pirating-software-choose-microsoft.ars

Morrisey, O., te Velde, D. W., & Adrian Hewitt. (2004). Definición de Bienes Públicos Internacionales: Elementos Conceptuales. En M. Ferroni & A. Mody (Eds.), *Bienes públicos internacionales: Incentivos, Medición y Financiamiento*, Economía Internacional (págs. 29-42). México, D.F.: Banco Interamericano de Desarrollo y Alfaomega Colombiana S. A. Recuperado a partir de http://www-wds.worldbank.org/external/default/WDSContentServer/WDSP/IB/2005/08/08/000011823_20050808110237/Rendered/PDF/248841bienes0publicos.pdf

Moseley, B. (2000, Mayo). Adobe Fights For Marketshare. *Folio: The Magazine for Magazine Management | Find Articles at BNET*. Recuperado Julio 26, 2010, a partir de http://findarticles.com/p/articles/mi_m3065/is_6_29/ai_62793570/

Murphy, M. (2003). *Utilizing Linux in the Enterprise Environment: Linux Gets Down to Real Business*. IBM America's Linux Team. Recuperado a partir de http://www.sco.com/company/legal/update/2003-08-21_Linux_by_IBM.pdf

Netcraft. (2009, Enero). Netcraft SSL Server Survey for Sites in January 2009. *Netcraft*. Recuperado Abril 9, 2010, a partir de http://news.netcraft.com/ssl-sample-report/CMatch/Cosdv_uk

Netcraft. (2010, Marzo). March 2010 Web Server Survey. *Netcraft*. Recuperado Abril 9, 2010, a partir de http://news.netcraft.com/archives/2010/03/17/march_2010_web_server_survey.html

Netscape, & Microsoft. (1999). *Microsoft Engaged In A Predatory Campaign To Crush The Browser Threat To Its Operating System Monopoly* (pág. 218). Recuperado a partir de http://www.justice.gov/atr/cases/f2600/v-a.pdf

Nonaka, I., & Teece, D. J. (2001). *Managing industrial knowledge*. London: SAGE.

Novell. (2010a). Linux Windows Integration. *Linux and Windows Interoperability*. Recuperado Abril 9, 2010, a partir de http://www.novell.com/linux/interoperability.html

Novell. (2010b). SUSE Linux Enterprise Desktop: File Formats for Office Applications. Recuperado Abril 9, 2010, a partir de http://www.novell.com/products/desktop/fileformats.html

Novell. (2010c). SUSE Linux Enterprise Consolidation Suite for System z for IBM Solution Edition for Enterprise Linux. Recuperado Abril 9, 2010, a partir de http://www.novell.com/products/systemz/els.html

Nyholm, J., Normann, L., frelle-Petersen, C., Riis, M., & Torsten, P. (2001). Innovation Policy in the Knowledge-based Economy—Can Theory Guide Policy Making? En D. Archibugi & L. Bengt-Åke (Eds.), *The Globalizing Learning Economy* (págs. 253-272). Oxford University Press.

OHA. (2010, Junio 1). Members. *Open Handset Alliance*. Recuperado Julio 27, 2010, a partir de http://www.openhandsetalliance.com/oha_members.html

Oiaga, M. (2007, Enero 10). Vista - a $6 Billion Dollars Operating System - The best billions Bill Gates has ever spent. *Softpedia*. Recuperado Marzo 25, 2010, a partir de http://news.softpedia.com/news/Vista-a-6-Billion-Dollars-Operating-System-44096.shtml

Oliver, A. (2010, Mayo 23). To Microsoft, Open Source means "Windows Encumbered". *Open Source Initiative*. Recuperado Mayo 24, 2010, a partir de http://opensource.org/Microsoft-Open-Source

Oliver, R., & González, L. (2008). Experiencias de asociación para la innovación entre pequeñas empresas: El modelo de integradoras de negocios en el sector de software en Jalisco. En *Transferencia del Conocimiento y la Tecnología: reto en la Economía Basada en el Conocimiento* (págs. 1-22). Presented at the Congreso Internacional de Sistemas de Innovación para la Competitividad Tercera Edición, León, Guanajuato: QUIVERA/CONCYTEG. Recuperado a partir de http://octi.guanajuato.gob.mx/sinnco/formulario/MT/MT2008/MT5/SESION3/MT5_OLIVER_GO NZALEZ.pdf

Oracle. (2010). About OpenOffice.org. *OpenOffice.org | The Free and Open Productivity Suite | About*. Recuperado Mayo 26, 2010, a partir de http://about.openoffice.org/index.html

Ordóñez, S. (2006, Enero). Capitalismo del conocimiento: elementos teórico-históricos. *Economía Informa*, *338*, 23-33.

Ordóñez, S., Correa, M., & Ortega, R. (2008). Capitalismo del conocimiento: alternativas de desarrollo nacional en el software libre y de fuente abierta. *Economía Informa*, *352*, 41-64.

Open Source Initiative (OSI). (2007, Octubre 12). OSI Approves Microsoft License Submissions. *Open Source Initiative*. Recuperado Mayo 31, 2010, a partir de http://opensource.org/node/207

Open Source Initiative (OSI). (2010). The GPL:Licensing. *Open Source Initiative*. Recuperado Mayo 13, 2010, a partir de http://www.opensource.org/licenses/gpl-2.0.php

Parlamento Europeo. (2004, Abril 28). DECISIÓN 2004/387/CE DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 21 de abril de 2004 relativa a la prestación interoperable de servicios paneuropeos de administración electrónica al sector público, las empresas y los ciudadanos (IDABC). *Diario Oficial de la Unión Europea*. Europa.

Phipps, S. (2010, Febrero 28). Time To Rebut The IIPA's FUD Against Open Source. *Open Source Initiative*. Recuperado Mayo 24, 2010, a partir de http://opensource.org/node/510

Pingdom. (2009, Abril 30). The money made by big tech companies like MS, Google, Apple, IBM and more. *Royal Pingdom: Covering Pingdom, Internet, technology and anything uptime-related*. Recuperado Abril 7, 2010, a partir de http://royal.pingdom.com/2009/04/30/the-money-made-by-big-tech-companies-like-ms-google-apple-ibm-and-more/

Polanyi, K. (2007). *La gran transformacion : Critica del liberalismo económico*. Madrid: La Piqueta/Quipu.

Ramos, A. A., Argott, L., & Barrueta, G. (2004). Hacia una metodología crítica en las ciencias sociales. En *Enfoques metodológicos críticos e investigación en ciencias sociales* (págs. 13-44). México: Plaza y Valdes.

Raymond, E. S. (1998, Noviembre 20). La Catedral y el Bazar. (J. Soto Pérez, Trad.). Recuperado Abril 6, 2010, a partir de http://biblioweb.sindominio.net/telematica/catedral.html

Raymond, E. S. (2000, Agosto 24). Homesteading the Noosphere. Recuperado Octubre 14, 2009, a partir de http://www.catb.org/~esr/writings/homesteading/homesteading/index.html

Red Hat. (2009a). *Red Hat Inc. Form 10-K (Annual Report)*. Raleigh North Carolina: EDGAR Online.

Red Hat. (2009b). *Contrato Red Hat Para Empresas*. México: Información Confidencial de Red Hat. Recuperado a partir de http://www.redhat.com/licenses/Enterprise_Agr_Mexico.pdf

Red Hat. (2010). Corporate Facts. *redhat.com*. Recuperado Abril 9, 2010, a partir de http://www.redhat.com/about/companyprofile/facts/

Revelli, M. (2004, Noviembre 27). 8 tesis sobre el postfordismo. *ediciones simbioticas*. Recuperado Febrero 20, 2010, a partir de

http://www.edicionessimbioticas.info/8-tesis-sobre-el-postfordismo

Richtel, M. (1998, Octubre 22). Memos Released in Sun-Microsoft Suit. *The New York Times*. Recuperado a partir de http://www.nytimes.com/1998/10/22/business/memos-released-in-sun-microsoft-suit.html?sec=&spon=&partner=permalink&exprod=permalink

Robert, V. (2004, Septiembre 1). Límites y potencialidades de la difusión de software libre en un país en desarrollo. El caso de la Argentina. Universidad Nacional de General Sarmiento.

Rodríguez, F. (2007). IT Complements amplía su red de consultores certificados. *Beneficios tangibles*. Recuperado Mayo 4, 2010, a partir de http://www.itcomplements.com/noticia_it/Noticias2008/ConsultoresCertificadosITC.htm

Romo, F. (2003, Septiembre 30). Entrevista con Fernando Romo - Vendemos el Misterio. Recuperado a partir de http://www.revista.unam.mx/vol.4/num5/entrev/entrevista1.htm

Rullani, E. (2000a). El valor del conocimiento. En *Territorio, conocimiento y competitividad de las empresas* (págs. 229-258). Madrid: Miño y Dávila.

Rullani, E. (2000b, Mayo). El capitalismo cognitivo: du déjà vu? (B. Baltza, Trad.)*Multitudes*, *2*. Recuperado a partir de http://www.sindominio.net/arkitzean/multitudes/multitudes2/rullani.htm

Rullani, E. (2000c, Mayo). Produccion de conocimiento y valor en el posfordismo. (B. Baltza, Trad.). Recuperado a partir de http://www.sindominio.net/arkitzean/multitudes/multitudes2/corsani-rullani.htm

Sampere, J. C., & Buenrostro, E. (2008). Building Policies for the Software Sector in Mexico: Regional Policies for a Global Industry (págs. 1-17). Presented at the Prime-Latin America Conference, Mexico City: Prime.

Sanchez, C. (2007, Marzo 22). MySQL Gains 25% Market Share of Database Usage, Latest Evans Data Survey Shows. *EDC | Press | Press Release*. Recuperado Mayo 27, 2010, a partir de http://www.evansdata.com/press/viewRelease.php

Sandler, T. (2004). Financiación de Bienes Públicos Internacionales. En M. Ferroni & A. Mody (Eds.), *Bienes públicos internacionales: Incentivos, Medición y Financiamiento*, Economía Internacional (págs. 75-110). México, D.F.: Banco Interamericano de Desarrollo y Alfaomega Colombiana S. A. Recuperado a partir de http://www-wds.worldbank.org/external/default/WDSContentServer/WDSP/IB/2005/0

8/08/000011823_20050808110237/Rendered/PDF/248841bienes0publicos.pdf

Schönitzer, M. F. (2008, Octubre 9). Linux Kernel Statistics! Recuperado Mayo 8, 2010, a partir de http://www.schoenitzer.de/lks/lks_en.html

Schumpeter, J. (1983). *Capitalismo, socialismo y democracia*. Biblioteca de Economía. Barcelona: Orbis.

Select. (2005, Febrero). *Alineando la tecnología a las oportunidades mexicanas. Tendencias 2005*. Presented at the XII Reunión Anual de Estrategias de Negocios, México.

Smith, M. A., & Kollock, P. (1999). Introduction. En *Communities in cyberspace* (págs. 3-25). Routledge.

Stallman, R. M. (2004). *Software libre para una sociedad libre*. (J. Rowan, D. Sanz Paratcha, & L. Trinidad, Trads.). Madrid: Traficantes de Sueños.

Stallman, R. M. (2007). Why “Free Software” is better than “Open Source”. *Gnu.org*. Recuperado Octubre 18, 2009, a partir de http://www.gnu.org/philosophy/free-software-for-freedom.html

Stallman, R. M. (2009). Lest CodePlex perplex. *Free Software Foundation*. Recuperado Mayo 31, 2010, a partir de http://www.fsf.org/blogs/rms/microsoft-codeplex-foundation

Stallman, R. M., Ming, W., Rendules, C., & Kembrew, M. (2007). *CONTRA EL COPYRIGHT*. Grafic Gold.

Stezano, F. A. (2009). *Redes ciencia-industria para la transferencia en México, Estados Unidos y Canadá. Regímenes institucionales y tecnológicos y mecanismos de intermediación.* (Doctorado). México: FLACSO, Sede Académica de México. Recuperado a partir de http://www.flacso.edu.mx/biblioiberoamericana/TEXT/DOCCS_VI_promocion_2006-2009/Stezano_F.pdf

Stobbs, G. A. (2000). *Software patents*. Aspen Publishers Online.

Subversion. (2010). Subversion FAQ. Recuperado Mayo 7, 2010, a partir de http://subversion.apache.org/faq.html

Tanenbaum, A. S. (2003). *Sistemas operativos modernos*. Pearson Educación.

Tate, C. (2000, Abril 22). Micro-Soft. *HistoryLink.org, the Free Online Encyclopedia of Washington State History*. Recuperado Noviembre 25, 2009, a partir de http://www.historylink.org/index.cfm?

DisplayPage=output.cfm&file_id=2294

The Gale Group. (2005). Oracle Corporation -- Company History. *International Directory of Company Histories*. Recuperado Mayo 4, 2010, a partir de http://www.fundinguniverse.com/company-histories/Oracle-Corporation-Company-History.html

Thompson, J. B. (1995). *The media and modernity*. Stanford University Press.

Thorsten, L. (2010, Febrero 24). Conclusion, data and numbers, outlook for 2.6.34 - The H Open Source: News and Features. Recuperado Mayo 8, 2010, a partir de http://www.h-online.com/open/features/Conclusion-data-and-numbers-outlook-for-2-6-34-933427.html

TLF. (2010). Members. *The Linux Foundation*. Recuperado Mayo 11, 2010, a partir de http://www.linuxfoundation.org/about/members

Top500.org. (2008, Abril). Operating system Family share for 11/2008, 2000-2007. *Top500 supercomputers*. Recuperado a partir de http://www.top500.org/stats/list/32/osfam

Torvalds, L. (2005, Octubre 30). git versus CVS (versus bk). Recuperado a partir de http://marc.info/?l=git&m=113072612805233&w=2

Torvalds, L. B. (1991, Agosto 26). What would you like to see most in minix? (Del autor, Trad.)*comp.os.minix*. Recuperado Noviembre 26, 2009, a partir de http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b?pli=1

Turner, M. S. V. (2006). *Microsoft Solutions Framework*. Redmond: Microsoft Press. Recuperado a partir de chm:file:///home/cano/P%C3%BAblico/0735623538.chm!/0735623538/toc.html

UNU-MERIT. (2006). *Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU*. Netherlands: European Commission. Recuperado a partir de : http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf

Updegrove, A. (2009, Septiembre 14). The CodePlex Foundation: First Impressions (and Recommendations). *ConsortiumInfo.org*. Recuperado Mayo 31, 2010, a partir de http://www.consortiuminfo.org/standardsblog/article.php?story=20090914102959510

Valloppillil, V. (1998, Agosto 11). Halloween Document 1. Recuperado Abril 6, 2010, a partir de http://www.catb.org/~esr/halloween/halloween1.html#quote7

Vidal, M. (2000, Julio). Cooperación sin mando: una introducción al software libre - Desafíos e interrogantes - Wikilearning. Recuperado Noviembre 8, 2009, a partir de http://biblioweb.sindominio.net/telematica/softlibre/sl.pdf

Von Hippel, E., & Von Krogh, G. (2002). Exploring the open source software phenomenon: issues for organization science. En *Annual meeting of the Academy of Management Meeting* (págs. 208-223). Denver, CO.

W3Counter. (2010, Febrero 28). Global Web Stats. *W3Counter*. Recuperado Marzo 25, 2010, a partir de http://www.w3counter.com/globalstats.php

Walker-Morgan, D. (2009, Julio 24). Microsoft and Vyatta rebut reports of GPL violation. *The H Open Source: News and Features*. Recuperado Mayo 31, 2010, a partir de http://www.h-online.com/open/news/item/Microsoft-and-Vyatta-rebut-reports-of-GPL-violation-742629.html

Walsh, I. (2006). What is QuarkXPress? *Technical Writing | Writing Tutorials | Free Word Templates*. Recuperado Julio 26, 2010, a partir de http://www.klariti.com/technical-writing/What-is-Quark-XPress.shtml

Weber, S. (2000). *The Political Economy of Open Source Software*. BRIE Working Papers Series. California: Alfred P. Sloan Foundation.

West, J. (2003). How open is open enough?:: Melding proprietary and open source platform strategies. *Research Policy*, *32*(7), 1259–1285.

Wheeler, D. A. (2005, Mayo 18). Comments on Open Source Software / Free Software (OSS/FS) Software Configuration Management (SCM) / Revision-Control Systems. Recuperado Mayo 7, 2010, a partir de http://www.dwheeler.com/essays/scm.html

Wheeler, D. A. (2007, Abril 16). Why Open Source Software / Free Software (OSS/FS, FOSS, or FLOSS)? Look at the Numbers! Recuperado Noviembre 27, 2009, a partir de http://www.dwheeler.com/oss_fs_why.html

Wikipedia, C. (2010, Marzo 1). VisiCalc. *Wikipedia, The Free Encyclopedia*. Recuperado Mayo 4, 2010, a partir de http://en.wikipedia.org/wiki/VisiCalc

World International Property Organization (WIPO). (2006). Intellectual Property Handbook: Policy, Law and Use. En *TECHNOLOGICAL AND LEGAL DEVELOPMENTS IN INTELLECTUAL PROPERTY,*. Recuperado a partir

de http://www.wipo.int/about-ip/en/iprm/pdf/ch7.pdf

World Information Technology and Services Alliance (WITSA). (2006). *Digital Planet 2006: The Global Information Economy*. Vienna, Virginia.

Yoguel, G. (2008). Información y conocimiento: las vinculaciones entre difusión de TIC y competencias tecnológicas. En V. Giovanna, M. Casalet, & A. Dante (Eds.), *Instituciones, sociedad del conocimiento y mundo del trabajo* (págs. 295-326). México: FLACSO sede México-Plaza y Valdés.

Yoguel, G., Robert, V., Analía, E., & José, B. (2006). Capacidades Cognitivas, Tecnologías y Mercados: de las Firmas Aisladas a las Redes de Conocimiento. En M. E. Albornoz & C. Alfaraz (Eds.), *Redes de conocimiento: construcción, dinámica y gestión* (págs. 37-63). Buenos Aires: Redes, Centro de Estudios sobre Ciencia, Desarrollo y Educación Superior.

Yoguel, G., Robert, V., Erbes, A., & Borello, J. (2005, Noviembre). Capacidades Cognitivas, Tecnologías y Mercados: de las Firmas Aisladas a las Redes de Conocimiento. Agencia Nacional de Promoción Científica y Tecnológica (Argentina). Recuperado a partir de http://ricyt.centroredes.mine.nu/ponencias/yoguel.pdf

Zemelman, H. (1992). *Los horizontes de la razón: Uso crítico de la teoría* (Vols. 1-2, Vol. 1). Anthropos Editorial.

Zemelman, H. (1996a). *Uso Crítico De La Teoría En Torno a Las Funciones Analíticas De La Totalidad*. México: Universidad de la naciones Unidas.

Zemelman, H. (1996b). *Problemas antropológicos y utópicos del conocimiento*. México: Colegio de México, Centro de Estudios Sociológicos.

Zúñiga, L., & Camacho, K. (2004). *Informe parcial de investigación Software Libre en América Latina y el Caribe*. Voces libres de los campos digitales. San José: Oficina Bellanet para América Latina y el Caribe

Glosario

- **Berkley Software Distribution (BSD)**: Primer sistema operativo desarrollado por la praxis de lo que ahora se conoce como software libre. También enarbola una licencia de software que permite restringir el acceso a las obras derivadas que ampara (págs. 32, 33, 46, 70-73, 73-75, 80, 82, 83, 92, 93, 95, 100, 165, 184, 194).
- **Bien club:** Tipo de bien que no puede ser catalogado ni como puramente público, ni como puramente privado, dado que cuenta con un mecanismo de exclusión. En el caso del software libre, este mecanismo se basa en el costo de adquirir la capacidad de su aprovechamiento, el cual se traduce en tiempo y esfuerzo por parte de aquel que lo desea comprender y apropiarlo como conocimiento (págs. 28, 29, 51-56, 64, 65, 84, 95, 122, 125, 137, 139, 141, 143, 144, 163, 165-168, 169, 186-190, 194, 198, 201, 206).
- **Bien inmaterial o intangible:** Es el producto de las competencias, habilidades y conocimiento de los trabajadores que lo producen y su cooperación, donde lo económico sólo puede normarlo y estandarizarlo. En el ámbito contable denomina a aquel bien que: a) se emplea continua y repetitivamente en la actividad del ente, b) tiene una capacidad de servicio que no se agota ni consume con el primer empleo y c) mientras está en uso no se transforma en otro(s) bien(es) ni está destinado a la venta (págs. 42-45, 51).
- **Bug:** Así se le llama en el lenguaje técnico a errores de programación (págs. 72, 83, 86, 90-91, 97, 104-106, 107, 123, 128, 128, 140, 166, 176, 180).
- **Código fuente**: Tiene la característica de poder ser leído por personas que conozcan el lenguaje de programación en que se ha empleado para desarrollarlo. Este código, posteriormente puede ser traducido en código legible por una computadora (código objeto), a través del uso de programas compiladores (págs. 12, 30, 31, 33, 55, 58, 71, 89, 111, 128,

133, 157, 166, 166, 172, 184).

- **Conocimiento:** Un proceso dinámico humano de justificar la creencia personal hacia la verdad, donde el contexto que le da sentido es la experiencia humana (véase: Conocimiento codificado y conocimiento objetivado y especialmente págs. 46-49).
- **Conocimiento codificado**: Aquel que, a partir de su formalización, es susceptible de ser apropiado como conocimiento. En este sentido el software libre es conocimiento codificado, ya que permite aprenderlo a cualquiera con conocimientos suficientes y tiempo. (págs. 1, 48, 50, 54, 63, 66, 70, 71, 93, 95, 102-103, 108, 112, 112, 115-116, 118, 122-124, 137, 139, 141-143, 144, 156, 160, 164, 167-169, 172, 177, 188-189, 194, 198-206).
- **Conocimiento objetivado (sintetizado, formalizado o concreto):** Es aquel que es plasmado en medios físicos y no puede ser fabricado, sino que debe ser desarrollado. Para ello, es necesaria la combinación fluida de experiencias, valores, informaciones contextuales y competencias especializadas. Todo lo anterior, da un cuadro de referencia a la evaluación y asimilación de nuevas experiencias y nuevas informaciones, que permiten decir que se crea y adapta conocimiento. El caso del software puede ser pensado como conocimiento sintetizado, concreto u objetivado, ya que se ha formalizado y plasmado en medios electrónicos para poder realizar su valor de uso, por lo que sus principales características son: ser lógico, no degradarle a través del tiempo y ya que se convierte en información puede ser reproducido sin esfuerzo y de manera multiplicativa. Puesto que tanto aquel que lo crea o lo posee en un primer momento como aquel que recibe una copia, tienen el mismo software. Pero, no es posible apropiarlo como conocimiento si no es posible comprender el porqué funciona (págs. 48, 49, 50, 53, 58, 63-64, 69, 93, 95, 101-102, 139, 156, 160, 194, 198, 206).
- **Destrucción creativa:** El dato de hecho esencial del capitalismo según la visión schumpeteriana, donde toda empresa que quiera sobrevivir debe

adaptarse a esta dinámica. Según ella, el proceso del capitalismo demanda que sus empresas continuamente creen nuevos bienes de consumo, nuevos métodos de producción y transporte, nuevas formas de organización industrial. Y al hacerlo, esta creación continua de elementos nuevos destruye lo que antes era dominante (págs. 21, 110, 120).

- **Doble Movimiento:** Es la acción de dos principios organizadores en el interior de la sociedad, cada uno de los cuales presenta específicos objetivos institucionales, cuenta con el apoyo de fuerzas sociales determinadas y emplea métodos propios (págs. 20-21 22, 24, 26, 27, 157, 177, 179-186, 189, 192, 194, 196, 206).
- **Drivers**: Programa de computadora que permite al sistema operativo utilizar algún hardware (págs. 77, 88, 128, 136, 172, 183).
- **Enterprise resource planning** (**ERP)**: Tiene por objetivo ser un software estandarizado capaz de controlar las distintas fases del proceso de negocios (págs. 67, 157, 163, 187).
- **Ganancia (plusvalor o plusvalía):** Es la diferencia entre el costo de producción de la mercancía y el precio de su cambio en el mercado (véase: Especialmente pág. 38)
- **Genaral Public License** (**GPL**): Es una licencia utilizan las herramientas legales creadas para proteger la obra intelectual de los autores, la cual ampara su facultad para decidir sobre sus obras. Por ello, la GPL, exige acceso al código fuente del software bajo la condición de que toda obra derivada deberá ser también liberada en los mismos o incluso más permisivos términos, y no podrá ser apropiada con la intención de restringir el acceso ella. Así nace la distinción de software libre (free software), como aquel software que permite libre acceso al código, modificarlo y redistribuirlo, de forma gratuita o a cambio del pago de un precio (págs. 31, 46, 74, 87, 92, 100, 128, 133, 137, 144, 163, 165, 181, 182-185, 190).
- **GNU No es UNIX (GNU)**: Acrónimo recursivo que designa el proyecto

fundado por Richar Stallman, para crear un sistema operativo libre (págs. 30-32, 58, 72, 73-76, 79, 80, 90, 92, 94, 129-131, 132, 136, 157, 159, 162).

- **Gran empresa:** véase: Pequeña y mediana empresa (PyME).
- **Hacker:** Un término extendido entre la comunidad científica que significa: "innovación, virtuosismo y estilo ético". La cultura hacker es en esencia, una cultura de convergencia entre los humanos y sus máquinas en un proceso de interacción sin trabas. Es una cultura de creatividad tecnológica basada en la libertad, la cooperación, la reciprocidad y la informalidad (págs. 32, 58, 71-72, 82, 90, 91, 97).
- **Hardware**: Comprende todos los componentes físicos (mecánicos, magnéticos, eléctricos y electrónicos, incluidos los elementos periféricos) de una computadora o de un sistema de procesadores (págs. 1, 7, 8, 3, 30, 66-68, 88, 95, 113, 128,132, 135, 148, 172, 173, 180, 186).
- **Kernel**: componente central del sistema operativo (págs. 12, 31, 32, 70, 73-76, 76-82, 87, 90, 123, 156).
- **Linux, Apache, MySQL, PHP y Perl (LAMPP):** Así se le llama a la combinación de programas más comúnmente usados para poner en funcionamiento un servidor basado en software libre (págs. 156, 187).
- **Malware**: Software producido con la intención de realizar tareas no autorizadas y potencialmente perjudiciales para los intereses de aquel que lo ha puesto en funcionamiento (aunque muchas veces sin saberlo). Este tipo de software incluye entre sus tipos más conocidos: virus de computadora, gusanos, troyanos, *spyware* y *adware* (págs. 89, 172, 176).
- **Monopolio**: Condición no deseada del mercado, en la que existe un sólo vendedor en algún punto de la cadena de producción, lo que anula la libre competencia. El monopolio, permite explotar la renta al máximo de una curva dada de demanda, la cual, también puede ser alterada de manera artificial por la introducción de publicidad que crea nuevas necesidades a los consumidores o bien, nuevas maneras de satisfacer

viejas necesidades (págs. 26, 39, 50, 68, 96, 101, 108, 109-113, 115, 116, 119-122, 129, 131-133, 141-142, 144, 176).

- **Open Source Initiative** (**OSI**): Es una organización que creó el término software de fuente abierta, o bien *open source software* (OSS), el cual postula como objetivo: evitar el término "libre", ya que en inglés el término "*free*", también significa gratis. Así, llama la atención del público consumidor sobre dos asuntos, el primero es hacer hincapié en que la principal característica del software que producen es el acceso al código fuente. Por otro lado, el segundo asunto, es que no todo el software de fuente abierta es proporcionado de manera gratuita (págs. 13, 33-35, 92, 182, 183).
- **Pequeña y mediana empresa (PyME):** En México normalmente se incluye a la micro empresa a menos que se indique lo contrario, por lo que se ha preferido al uso del acrónimo MiPyME. En este país se determina la categoría de acuerdo al número de empleados por el sector de la economía en el que se desempeña: Industria, comercio o servicios. Empero se considera microempresa a aquella que va de 0 a 10 empleados en cualquier caso. Por otro lado, se considera pequeña empresa a aquella que ocupa de 11 a 50 personas en el ramo de servicios e industria, y aquella que tiene de 11 a 30 en el sector de comercio. Por último, se considera mediana empresa a aquella que tiene de 21 a 100 empleados en los ramos de industria y servicios o bien, si se encuentra en la parte de comercio, a aquella que va de 31 a 100 trabajadores. Son consideradas grandes empresas las que tienen más de 100 empleados en todos los casos (págs. 155, 159, 160, 162-163, 165, 174, 177, 179-180, 187-188, 193, 205, 206-207, 208).
- **Plugins**: Piezas de software que permiten a distintas tecnologías de software interactuar unas con otras (págs.88, 172).
- **Práctica Social:** Constituye un producto y un ámbito de conflicto de entre fuerzas sociales que luchan por imponer sus proyectos, de acuerdo con la conciencia que tengan sus intereses. Toda práctica social conecta

pasado y futuro en su concreción presente, ya que siempre se mostrará una doble subjetividad: como reconstrucción del pasado (memoria) y como apropiación del futuro, dependiendo la constitución del sujeto de la articulación de ambas (págs. 19-22, 23, 28, 29, 51, 100, 144).

- **Script**: En informática, se refiere a un conjunto de instrucciones que permiten automatizar una tarea (págs. 83, 104, 172).
- **Sistema operativo**: Software que permite comunicar el hardware (parte física) de la computadora con software que tiene un fin específico (aplicaciones). Por ejemplo: el teclado con un procesador de textos (págs. 30-32, 58, 67-70, 70-73, 73-76, 81, 87, 110-112, 114-115, 119-122, 128, 129-133, 134, 137 138, 142, 147, 149, 174, 176, 178, 180).
- **Software:** Conocimiento codificado en forma de texto, que al ser hecho concreto en medios electrónico-informáticos y procesado, permite que las computadoras funcionen (véase: Software libre y software propietario).
- **Software libre**: Bajo la concepción de la GPL, enarbola el derecho a que todos tengan acceso a modificar y redistribuir software, pero donde a ningún distribuidor se le permite restringir su redistribución posterior. Es decir, no están permitidas modificaciones propietarias, por lo que no se le puede considerar de dominio público. A pesar de ello, bajo la percepción de la BSD, se permite el derecho a que todos tengan acceso a modificar y redistribuir software e incluso a que aquel que lo modifica restrinja su acceso (véase: GPL y BSD).
- **Software propietario (o privado):** Prohíbe el acceso al código fuente y por lo tanto a la capacidad de estudiar el software, modificarlo y redistribuirlo. Es normalmente distribuido con una licencia de uso para el usuario final, donde la propiedad del código permanece con aquel que detenta los derechos de autor o *copyright* y/o, dependiendo del país, también patentes (véase: Software libre).
- **Tecnología:** En la concepción marxiana, es la ciencia aplicada al proceso mismo del trajo, es decir, tanto la destreza del trabajador como los

medios materiales de la producción (conformados por una cosa o conjunto de cosas que el trabajador interpone entre él y el objeto de trabajo, el cual, le sirve de vehículo de su acción sobre dicho objeto), en la realización de una actividad orientada a un fin, satisfacer una necesidad (véase: Especialmente págs. 37-38, 42, 56, 61, 61, 68, 113, 199)

- **Valorización:** En la acepción clásica marxista, es el resultado de la realización de una actividad orientada a un fin (trabajo), contenido en una mercancía producida para crear valores de uso y cuyo costo ha sido inferior al precio por el que se cambia en el mercado. Aquí, se entiende que la valorización en el capitalismo cognitivo puede ser normada y estandarizada, y puesto que sólo puede ser producida por formas de vida, se presenta en multitud de caminos posibles. Sin embargo, es factible identificar este proceso en el momento que el conocimiento se transforma para crear valor de uso (véase: Especialmente págs. 38, 43, 50, 106).